THE DETECTION OF FRAUDULENT TRANSACTIONS USING MACHINE LEARNING

A THESIS SUBMITTED TO
THE FACULTY OF ACHITECTURE AND ENGINEERING
OF
EPOKA UNIVERSITY

BY

ANISA GJONI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF THE MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

MARCH, 2024

Approval sheet of the Thesis

This is to certify that we have read this thesis entitled **“The detection of fraudulent transactions using Machine Learning”** and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Arban Uka
Head of Department
Date: March 01, 2024

Examining Committee Members:

Prof. Dr. Betim Çiço	(Computer Engineering)	________________
Prof. Dr. Bekir Karlik	(Computer Engineering)	________________
Assoc. Prof. Dr. Dimitrios A. Karras	(Computer Engineering)	________________

ABSTRACT

THE DETECTION OF FRAUDULENT TRANSACTIONS USING MACHINE LEARNING

Gjoni, Anisa

M.Sc., Department of Computer Engineering

Supervisor: Assoc.Prof. Dr. Dimitrios Karras

Credit cards are massively used nowadays for internet transactions performed at any moment, given that they have offered facilitation both in usage and time. With the growing usage of credit cards, there has also been an increase in their misuse capacity. Credit card deceits cause considerable financial loss not only for their owners but also for the financial companies.

The main objective of this research study is the identification of the fraudulence cases which may include the access of the public data, handling groups of largely destabilized data and the adaption to the developing deception models. The corresponding literature poses many approaches based on Machine Learning for the detection of credit cards, some of which are: Extreme Learning Method, Decision Tree, 0Random Forest, Support Vector Machine, Logistic Regression and XG Boost.

However, due to an insufficient accuracy, there is still some need to apply deeper algorithms to reduce the loss from fraudulence. For this aim, the main focus of this research wok has been the application of “Deep Learning” algorithms. A comparing analysis between the two algorithms “Machine Learning” and “Deep Learning” was conducted in order to retrieve efficient results. Also, a Machine Learning algorithm was applied on the group of data, which improved significantly the accuracy of detecting fraudulence. Moreover, I applied three architectures based on a convolutional neural network to ameliorate even further the performance of fraud detection. A complete empirical analysis was performed by experimenting with different configurations of the hidden layers by changing the number of training epochs and using the latest models.

The findings from this research demonstrate enhanced results, specifically in terms of accuracy and precision. The suggested model outperforms the most recent Machine Learning and Deep Learning algorithms designed for addressing credit card fraud detection issues. In addition, I conducted experiments to balance the data and implemented Deep Learning algorithms to reduce the occurrence of biased negative results. These proposed methods can be efficiently employed to identify instances of credit card fraud in real-world scenarios.

Keywords*: Credit Card Fraud Detection, Deep Learning, Machine Learning, Cybersecurity*

ABSTRAKT

DEDEKTIMI I TRANSAKSIONEVE MASHTRUESE NEPERMJET PERDORIMIT TE MACHINE LEARNING

Gjoni, Anisa

Master Shkencor, Departamenti I Inxhinierisë Kompjuterike

Udhëheqësi: Assoc.Prof. Dr. Dimitrios Karras

Njerëzit i përdorin kartat e kreditit për transaksione në internet në çdo moment, pasi ofrojnë një lehtësim në kohë dhe në përdorim. Me rritjen e përdorimit të kartave të kreditit, është rritur edhe kapaciteti i keqpërdorimit të tyre. Mashtrimet me kartat e kreditit shkaktojnë humbje të konsiderueshme financiare, si për mbajtësit e kartave të kredititashtuedhepërkompanitëfinanciare.

Objektivi kryesor i këtij studimi kërkimor është identifikimi i rasteve të mashtrimit, të cilat mund të përfshijnë aksesimin e të dhënave publike, trajtimin e grupeve të të dhënave shumë të çekuilibruara dhe përshtatjen ndaj modeleve në zhvillim të mashtrimit. Literaturat përkatëse paraqesin shumë qasje të bazuar në Machine Learning për zbulimin e kartave të kreditit, siç janë Metoda Extreme Learning, Decision Tree, Random Forest, Support Vector Machine, Logistic Regression dhe XG Boost.

Megjithatë, për shkak të saktësisë së pamjaftueshme, ekziston ende nevoja për të aplikuar algoritme më të thellë për të reduktuar humbjet nga mashtrimi. Fokusi kryesor ka qenë zbatimi i algoritmeve “Deep Learning” për këtë qëllim. Analiza krahasuese e të dy algoritmeve “Machine Learning” dhe të "Deep Learning" u krye për të gjetur rezultate efikase. Analiza e detajuar empirike është kryer duke përdorur të dhënat e standardeve evropiane të kartave për zbulimin e mashtrimit. Një algoritëm "Machine Learning" u aplikua në grupin e të dhënave, gjë që përmirësoi saktësinë e zbulimit të mashtrimeve në një masë të caktuar. Më pas, janë aplikuar tre arkitektura të bazuar në një rrjet nervor konvolucional për të përmirësuar performancën e zbulimit të mashtrimit. Shtimi i shtresave të thellësuara më tej rriti saktësinë e zbulimit.

Vlerësimi i punës kërkimore tregon rezultate të përmirësuara, si saktësia dhe precizioni. Modeli i propozuar tejkalon algoritmet më të fundit të mësimit të makinave dhe të mësimit të thellë për problemet e zbulimit të kartave të kreditit. Përveç kësaj, kemi bërë eksperimente duke balancuar të dhënat dhe duke aplikuar algoritme të mësimit të thellë për të minimizuar shkallën e rezultateve të gabuara negative. Qasjet e propozuara mund të zbatohen me efikasitet për zbulimin e mashtrimit me karta krediti në botën reale.

Fjalë kyçe: *Zbulimi i mashtrimeve me kartat e creditit, Deep Learning, Machine Learning, CyberSecurity*.

ACKNOWLEDGEMENTS

I would like to express my thanks and my gratitude to my supervisor, faculty, family, friends and all those who have supported me and believed in me throughout this journey. Your guidance, encouragement and cooperation have been invaluable to me and I am very excited to apply the knowledge and skills that I have gained to make meaningful contributions in the scientific field. Thank you for being a part of my success!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ML	Machine Learning
DL	Deep Learning
CCFD	Credit card fraud detection
ANN	Artificial Neural Network
RGB	Color model
CNN	Convolutional Neural Network
MSE	Mean Square Error
SVM	Support Vector Machine

CHAPTER 1

INTRODUCTION

In an era characterized by a rapid technology integration in various aspects of the everyday life, there has never been a more critical need for well-run data systems. With the proliferation of online transactions and expansion of digital payment systems, there is a proclivity in the need for more robust and adaptive fraud detection systems.

This master thesis offers a detailed exploration and analysis regarding credit cards fraud detection, with its main focus on the exploitation of advanced technologies, particularly Machine Learning and Deep Learning. The subsequent chapters present a sheer paradigm of the design and implementation of a system, whose main goal is the protection of financial transactions against a nefarious use. These approaches are taken as an attempt to thwart hackers from tampering with sensitive financial information and to build an unimpaired system that does not disclose any confidential data.

A systematic analysis has been performed for both existing systems used in the field and the proposed system, aiming to improve the shortcomings of the current methodologies. Chapter 2 concenters in the main requirements specifications, including the requests of hardware and software, and also the technologies used in the project, such as Python and ML, with particular emphasis on their relevance to credit card fraud detection.

Thence the system design and implementation are examined, by taking into account limitations in analysis and design and the critical security and performance requirements needed to optimally combat this evolving threat. Chapter 4 explains the system's architectural framework and it provides a visual representation, including the architecture diagrams and UML diagrams. Furthermore, this paper casts some light on the intricacies on the collection and processing of the data set. Lastly, the procedures of coding and testing are detailed, in the sense of coding standards, system testing protocols and various testing techniques that are critical to ensure the reliability and accuracy of fraud detection.

The testing phase offers recommendations to ameliorate the effectiveness of the system and a general overview of its impact in the credit card fraud detection problem. As a result, this paper will offer a better understanding of the various nature of threats posed to credit card usage, along with a practical overview of how the DL and ML technologies can be exploited to undermine the dangers related to financial transactions.

1.1. Methodology

The proposed paradigm serves to detect and improve credit card transaction data so as to build a system that securely retains such information and utterly separates it from illicit, harmful and deceitful transactions. I have taken my dataset from kaggle.com website and what is aimed through this research work is the annihilation of fraudulent transactions from this dataset. The built model is established using Deep Learning and Machine Learning in Python through tensor flow libraries and DL algorithms. I conducted my study with different operating systems, such as: Mac, Windows and Linux.

Dataset: In the dataset there is a total of 280.000 transactions. So, the dataset is a numerical one and it contains these variables: “time”, “amount”, “class” and also V1-V27 variables, which were created to store the identity of each of the users that have made these transactions.

Configurations: The programming language that I have used is Python (version 3.2). There have also been used Jupyter Notebook, Anaconda, TensorFlow and also Python libraries such as: KERAS (which allows the definition and training of neural network models), Pandas (used for analyzing, exploring and training the data), Numpy (used to work with arrows) etc.

Libraries: In my project of detecting fraudulent transactions, I have made use of several programming and data analysis libraries. Following is a more detailed description of them.

TensorFlow is an open-source machine learning software library created by Google, and I have employed it for constructing machine learning models.

Keras is an API for ML that integrates with TensorFlow and was used to facilitate the creation and implementation of ML models.

Pandas is a data processing library used to import, manipulate and analyze data. I used this library to read and manipulate Kaggle's credit transaction dataset.

NumPy is a library of Python used for scientific computation to manipulate data in the form of matrices and vectors. In this model it became useful for numerical data manipulation.

Seaborn is a data visualization library that is constructed upon Matplotlib. It served as a tool for representing data through visual elements such as graphics and various types of diagrams.

Matplotlib, on the other hand, is a comprehensive Python library that facilitates the generation of static and animated graphs, as well as diverse forms of data visualizations.

Scikit-Learn (sklearn) is a Python machine learning library that furnishes a variety of tools essential for training, evaluating, and validating machine learning models. In my applications, I utilized the “train_test_split” function to partition the data into training and testing sets. Additionally, I employed “StandardScaler” for normalizing the data.

The Confusion Matrix serves as a tool for assessing the effectiveness of a machine learning model. It shows how accurate or not the model’s classifications are.

Followed steps:

1. Retrieving the credit card transaction dataset from Kaggle. This dataset is a numerical one consisting from credit card transactions.
2. Exploring the data: Refers to the exploration of the dataset. I looked into the structure of the dataset, feature analysis and key data identification. Pandas, NumPy, Seaborn and Matplotlib were the libraries used for data analysis and visualization.
3. Data processing: After their exploration, I processed the data. This encompassed tasks such as cleaning empty data, addressing missing values, normalizing the data, and partitioning the dataset into training and testing sets. For this phase I used the Scikit-Learn library.
4. Modelling: After processing the data, the model was built. TensorFlow and Keras came in hand to build the ML model. I defined the model, selected the optimizer and compiled the model.
5. Model training: The training of the model was conducted using the training dataset. I used the optimization algorithms to update the model weights through the training data. Our model learns from the training set.
6. Model Evaluation: Following the training phase, an evaluation of its performance was conducted. Model evaluation methods were employed to gauge its effectiveness in accurately identifying fraudulent transactions.

1.2. Existing system

The pertinent literatures present various Machine Learning approaches for credit card fraud detection, including methods such as Extreme Learning Method, Decision Tree, Random Forest, Support Vector Machine, Logistic Regression, and XG Boost. However, these methods have been associated with low accuracy rates. In the year 2020 alone, there were approximately 393,207 reported cases of credit card fraud out of around 1.4 million identity theft reports. Starting from that year, Credit Card Fraud (CCF) emerged as the second most prevalent type of identity theft, closely following government document frauds and kleptocracy [1] . The global economy incurred a staggering cost of 24.26 million dollars due to card theft in the previous year. With credit card fraud accounting for 38.6% of reported losses in 2018, it is evident that the United States is particularly susceptible to identity theft [2].

1.3. The proposed system

DL algorithms are applied everywhere: in computer networks, banks, mobiles, medical discoveries, malware, location tracking etc. In this model I seek to identify thefts happening with credit cards in the banking institutions. A number of DL algorithms is used to identify CCF, however in this model I have decided to choose CNN model and its layers to detect the thefts, as well as the normal transactions in the set of data.

To address the imbalance in CCF data, a transformation was applied to create a balanced dataset by excluding non-fraudulent transactions from the original dataset.

The Convolutional Neural Network (CNN) model, featuring a layered architecture, was then employed on this balanced dataset to assess the proposed model. The CNN layers demonstrated a sequential architecture that resulted in training and validation accuracy exceeding 90%.

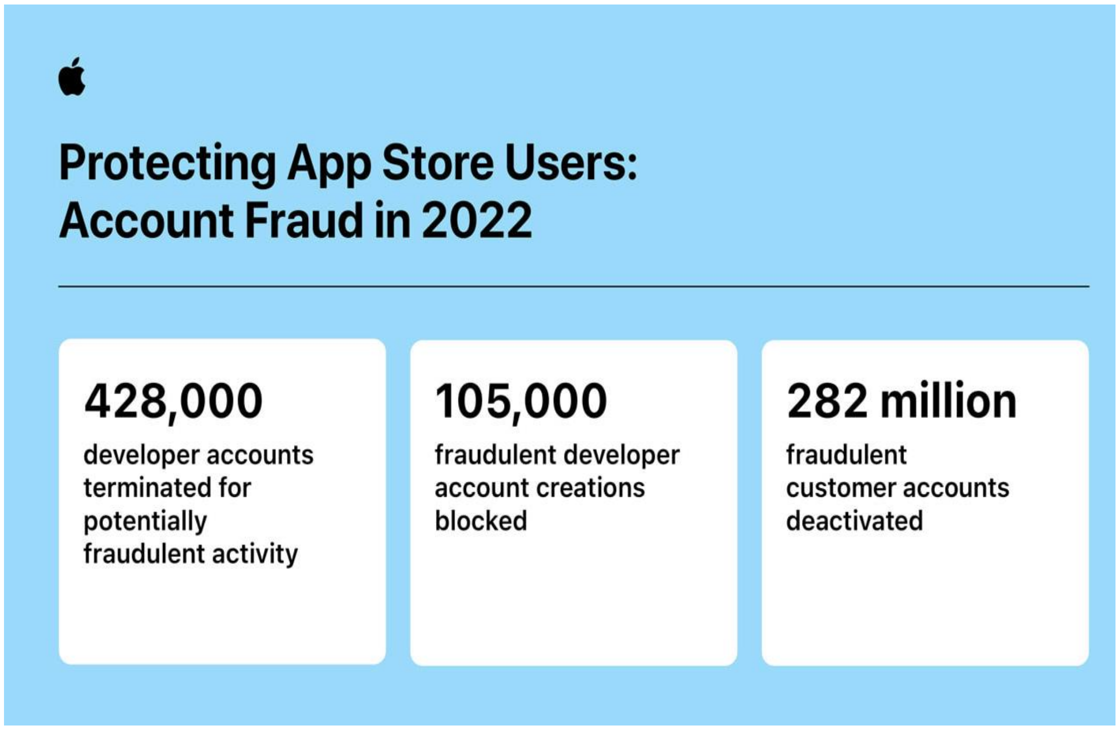
CHAPTER 2

REQUIREMENT SPECIFICATIONS

2.1. Credit card fraud

Credit card fraud (CCF) involves the unauthorized use of credit card or account details by someone other than the rightful owner, leading to illicit transactions. Fraud can occur when a credit card is lost, stolen, or forged. Additionally, fraudulent activities can take place even without the physical presence of a card, such as through the use of credit card numbers in e-commerce transactions, a trend that has become more prevalent with the rise of online shopping [3]. The surge in fraud cases, including CCF, can be attributed to the growth of e-banking and various online payment platforms.

In the current era dominated by online payments, detecting CCF has become a crucial objective. This is particularly significant as society will gradually transition toward a cashless culture. With the decline of traditional payment methods, businesses must adapt to these changes to remain relevant. Incentives, such as premiums for credit and debit card payments, are encouraging customers to move away from cash transactions. Consequently, companies need to update their financial systems to accommodate a variety of payment methods. The anticipation is that this situation will become more critical in the coming years.


Protecting App Store Users:
Account Fraud in 2022

428,000
developer accounts terminated for potentially fraudulent activity

105,000
fraudulent developer account creations blocked

282 million
fraudulent customer accounts deactivated

Figure 1.Apple's Fraudulent Statistics in 2022 [Apple.com]

The goal of supervised CCFD is to establish a Machine Learning model using historical credit card payment data. This model should possess the capability to distinguish between fraudulent and legitimate transactions, enabling it to assess the authenticity of incoming transactions. Critical considerations in this endeavor include the system's response time, cost sensitivity, and feature preprocessing [4] .

Machine Learning (ML), within the realm of artificial intelligence, involves utilizing computers to make predictions based on patterns observed in historical data. In the context of CCFD, the application of ML is pivotal for developing an effective and accurate system that can identify and mitigate fraudulent transactions.

2.2.1. Hardware requirements

- Hard Disc: 500GB
- RAM: 4 GB and above
- Processor: I3 and above

2.2.2. Software requirements

- Operating System: Windows 10 (64 bit)
- Software: Python
- Tools: Anaconda

2. 3. Used technology

- Python
- Deep Learning

2.3.1. Python

Python is a high-level, general-purpose programming language that enjoys widespread usage. Designed with a focus on simplicity, its syntax allows programmers to express concepts in a concise and clear manner.

One of Python's notable advantages is its ability to facilitate a rapid coding process, allowing developers to achieve more in fewer lines of code. This characteristic not only enhances the efficiency of coding but also promotes seamless system integration. Overall, Python stands out as a versatile and user-friendly programming language.

Utilization in Various Fields:

- Employed in web development, specifically from the server side.
- Widely used in software development processes.
- Applied in mathematical computations and analyses.
- Utilized for system scripting purposes.

Python applications:

- Python finds utility in server environments for building web applications.
- It has the capability to create workflows when combined with suitable software.
- Python is adept at interfacing with database systems and can proficiently read and modify files.
- Handling large datasets and executing intricate mathematical operations are among Python's strengths.
- It serves purposes ranging from rapid prototyping to the development of production-ready software.

Reasons for choosing Python:

- Python exhibits cross-platform compatibility, running seamlessly on Windows, Mac, Linux, and even Raspberry Pi.

- Its syntax is straightforward, resembling the English language, making it user-friendly for developers.
- Python's syntax efficiency enables the creation of programs with fewer lines compared to other programming languages.
- Python operates within an interpreter system, facilitating immediate code execution and expediting the prototyping process.
- Distinguishing itself from other languages, Python concludes commands with new lines instead of semicolons or parentheses.

Python is interpreted:

- Python is one of the most popular interpreted languages, meaning that it runs the code line by line and the command is carried out without first translating the source code into machine code.
- This allows for a faster development cycle, since the code is directly written and executed with no intermediate steps. Even if there is an error at the bottom of the code it will still produce an output up until the line of the program is correct and then it will stop and generate and error statement.
- One potential drawback of interpreted languages lies in execution speed, with programs compiled in the native language of the computer processor typically running faster than interpreted ones.
- Despite its syntactical simplicity, Python accommodates constructs anticipated by a high-level language. This includes support for complex and dynamic data types, structured and functional programming, as well as object-oriented programming.

2.3.2. What is Deep Learning

Deep Learning stands out as a specialized branch of Machine Learning, employing supervised, unsupervised, or semi-supervised learning techniques to glean insights from data representations. Its architecture bears a striking resemblance to the intricate workings of the human nervous system, featuring a complex network of interconnected computing units that collaborate harmoniously to process intricate information [5]. It's worth noting that Machine Learning operates on the premise that machines can acquire knowledge from data. Deep Learning, in turn, elevates Machine Learning to a more advanced and sophisticated level.

Deep Learning encompasses various facets, including:

- Multiple levels of hierarchical layouts
- Neural networks with many layers
- The training of expansive neural networks.
- Application of multiple nonlinear transformations.
- Proficiency in pattern recognition.
- Capability for feature extraction.
- Establishment of high-level data abstraction models.

An artificial neural network (ANN) represents a technique within artificial intelligence aimed at guiding computers to process data in a manner reminiscent of the human brain. Positioned as a subset of machine learning, ANN serves as the foundational framework for deep learning. It relies on a layered structure comprising interconnected neurons or nodes, closely mirroring the intricate organization of the human brain [6]. ANN is composed of four main parts, being: neurons, nodes, input and output.

Neurons

Artificial neural networks are structured with layers of neurons, where each neuron functions as a computational unit, processing information through weighted input parameters. The neurons individually weigh the inputs, sum them up, and then pass the result through a nonlinear function to generate the output. Each layer of neurons is adept at detecting specific information, such as identifying edges in images or locating tumors in the human body. Employing multiple layers of neurons enables the network to uncover additional insights about the input parameters.

Nodes

An artificial neural network is a network of interconnected nodes, resembling the layered structure of neurons in the brain. Each circular node signifies an artificial neuron, and arrows represent connections from the output of one neuron to the input of another. Inputs are initially transmitted to the first layer, where individual neurons receive specific input values. Subsequently, these values are used to compute a product based on their respective weights and interactions.

Results

The outcomes from the first layer are sequentially forwarded to the second layer for further processing. This iterative process continues until the final output is generated. After each iteration of passing data through the network, the resulting output is compared to the correct one, and adjustments are made to the values of the nodes until the network consistently produces the correct final output. This iterative learning process enhances the network's ability to approximate desired outputs over time.

2.3.2.1. Convolutional Neural Network (CNN)

The pivotal moment for neural networks occurred in 2012 during the annual computer vision Olympics, when CNNs were employed, reducing the classification error rate from 26% to an impressive 15%. This marked a significant improvement at that time and sparked widespread interest in neural networks [7]. Since then, numerous companies have incorporated machine learning (ML) at the core of their services. Prominent examples include Facebook utilizing neural networks for auto-tagging algorithms, Google relying on them for image search, Amazon employing them in product recommendations, Pinterest leveraging neural networks for home feed personalization, and Instagram integrating them into their search infrastructure [8]. While CNNs find application in various domains, their classic and most popular use case is often associated with image processing, particularly in image classification.

2.3.2.2. Image classification

Image classification refers to the process of taking an input image and determining its class or providing a probability distribution across various classes that best characterize the image. For humans, this cognitive ability is one of the earliest skills acquired, developing naturally and effortlessly from birth and becoming second nature as adults. Without conscious effort, we possess the capability to swiftly identify our surroundings and the objects within them. These abilities, such as rapid pattern recognition, generalization from prior knowledge, and adaptation to diverse image environments, set us apart from machines. The innate capacity to quickly recognize patterns and make sense of visual information underscores the human ability to navigate and understand the visual world.

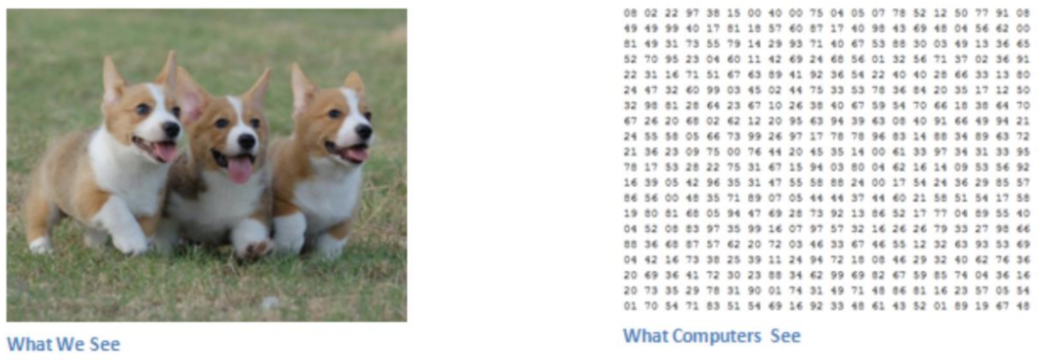


Figure 2. CNN image classification [Neural Network from scratch, Victor Zhou (2007)]

Input and Output

When a computer processes an image as input, it perceives it as an array of pixel values. The size of this array is determined by the resolution and dimensions of the image, often represented as a set of numbers such as 32 x 32 x 3, denoting RGB values. For instance, in the case of a colored JPG image with dimensions 480 x 480, the corresponding array would be 480 x 480 x 3. Each numerical value in this array, ranging from 0 to 255, signifies the intensity of the pixels at that specific point. While these numerical values might seem arbitrary to us, they constitute the sole input available to the computer.

In essence, the primary concept involves presenting the computer with this array of numbers, and in return, it produces numerical outputs that indicate the probability of the image belonging to a particular class. The computer, through the process of image classification, translates pixel information into meaningful predictions about the content or category of the given image.

2.3.2.3 Biological connection

The foundational principles of CNNs, draw inspiration from the workings of the visual cortex in the human brain. The visual cortex comprises small cell areas that exhibit sensitivity to specific regions of the visual field. This concept traces back to a pivotal experiment conducted in 1962, where it was unveiled that individual neuronal cells in the brain exhibited distinct responses to the presence of edges with specific orientations. For instance, certain neurons reacted to vertical edges, while others responded to horizontal or diagonal edges [9].

The manner in which CNN works is that it starts with an input image, which is a grid of pixel values. Then it goes through convolutional layers, nonlinear activation, pooling layers, fully connected layers and the output one.

2.3.2.4. First Layer – Convolutional layer

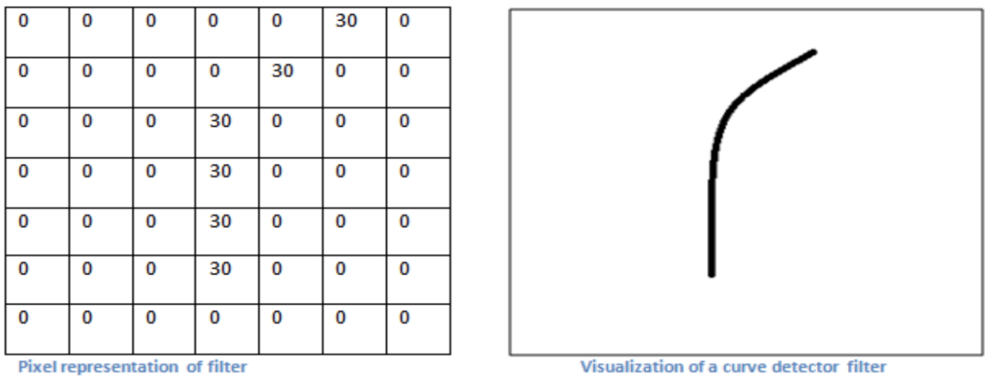
In a Convolutional Neural Network (CNN), the first layer is typically the convolutional layer, responsible for processing the input, which is a collection of pixel values representing an image. Convolution involves the movement of a filter (or kernel) across the image, conducting mathematical operations to extract features. Each convolutional layer captures distinct aspects and patterns of the input image.

To grasp how a convolutional layer works, we imagine a flashlight shining in the upper left corner of the image, covering a 5 x 5 area. This flashlight that serves as a filter or kernel, slides across the entire image, and the area it illuminates is referred to as the receptive field. The filter is essentially a set of numbers called weights or parameters, and its depth must match the depth of the input; for instance, a 5 x 5 x 3 filter for a colored image (RGB channels).

At each position, the filter multiplies its values by the corresponding pixel values of the image, and the results are summed up to produce a single representative value. This process repeats as the filter moves right, pixel by pixel, until it covers the entire image. Each position of the filter produces a value, and collectively, these values form a 28 x 28 x 1 activation map or feature map. The size of 28 x 28 is derived from the fact that a 5 x 5 filter can fit into a 32 x 32 input image in 784 different positions. If multiple filters, such as two 5 x 5 x 3 filters, are used, the output volume becomes 28 x 28 x 2. Employing more filters enables the CNN to recognize more complex features than when using a smaller number. This flexibility allows the network to identify not only edges or simple shapes but also more intricate patterns within the input image.

First layer – High level perspective

When discussing simple characteristics that are common to all images, such as straight edges, solid colors, and curves, each applied filter in a CNN can be viewed as a feature identifier. For instance, let's consider a hypothetical scenario where the first filter is designed to be a 7 x 7 x 3 curve detector. In the context of being a curve detector, this filter is essentially a pixel structure with higher numerical values positioned in an arrangement that corresponds to the shape of a curve.



0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter
Visualization of a curve detector filter

Figure 3. Curve detector filter [The Artificial Neural Networks handbook, 2018]

As the filter slides across the image during the convolution process, it identifies and responds to regions in the image that exhibit a curve-like pattern. The areas with the highest numerical values in the filter align with the presence of curves in the input image. When this filter is utilized in the upper left corner of an image, it aggregates the results of the multiplications between the values of the filter and the corresponding pixel values of the image within that specific area.

To illustrate, let's consider an image earmarked for classification, and we position the filter in the upper left corner of this image.

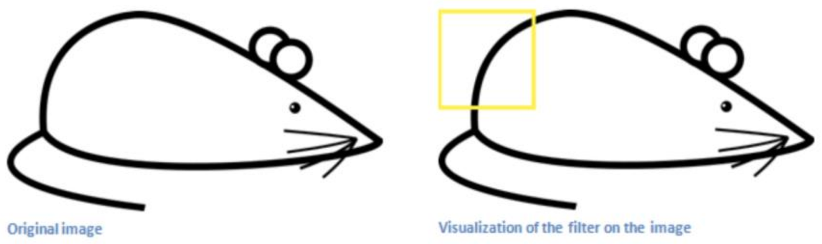
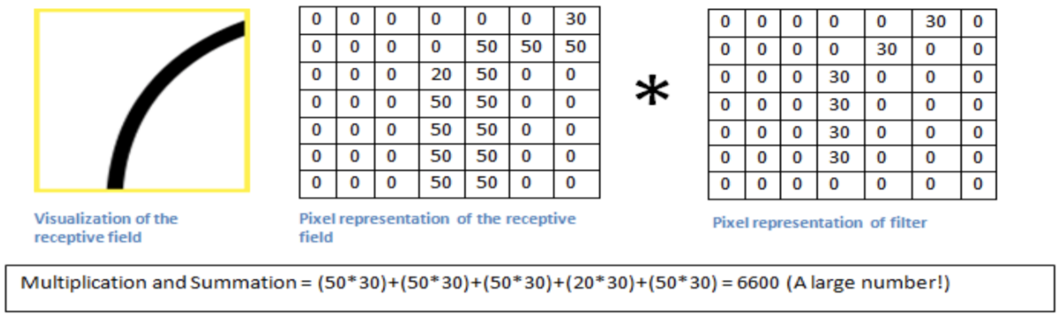


Figure 4. Visualizing the filter on the image [The Artificial Neural Networks handbook, 2018]

To classify the image, the essential step involves multiplying the values of the filter by the original pixel values of the image.


Visualization of the receptive field
Pixel representation of the receptive field
*
Pixel representation of filter

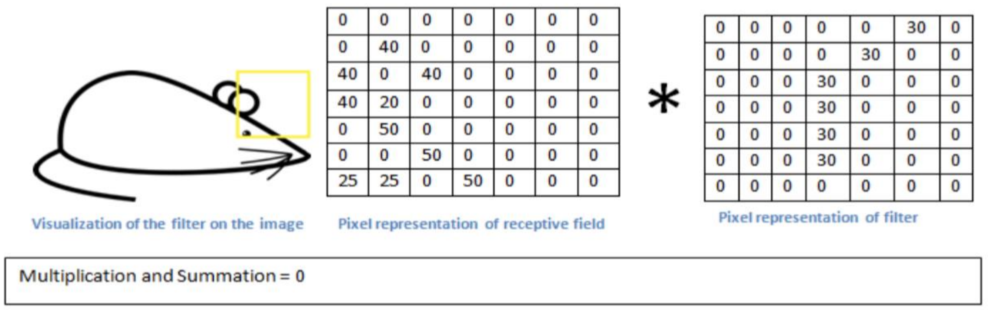
0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Figure 5. Multiplication and summation of filter values [The Artificial Neural Networks handbook, 2018]

In essence, in the input image, if there exists a shape resembling the curve represented by that filter, the total sum of the multiplications will yield a substantial value.


Visualization of the filter on the image
Pixel representation of receptive field
*
Pixel representation of filter

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Multiplication and Summation = 0

Figure 6.Image shape not resembling the filter curve [The Artificial Neural Networks handbook, 2018]

As evident, the value is notably low, indicating that there was no corresponding pattern in the image section for the curve detector filter. The output of this convolutional layer manifests as an activation map. In a simple scenario like rotation using a filter, the activation map reveals areas in the image where curves are likely present. In this specific example, the upper left value in the 26 x 26 x 1 activation map (due to a 7 x 7 filter instead of 5 x 5) is 6600. This high value suggests the probable existence of a curve in the image that triggered the filter. Conversely, the value on the right of the activation map is 0, indicating the absence of curves in that part of the original image, preventing the filter from activation.

This example involves only one filter designed to detect lines curving outward and to the right. To enhance feature detection, additional filters can be introduced, which increment the depth of the activation map, providing more comprehensive information about the input volume.

Denial

The filter I elucidated in this section was simplified primarily to illustrate the processes involved in convolution. In the image below, I will exemplify actual visualizations of the first layer convolution filters in a trained network. These filters in the initial layer traverse around the input image and become activated when they identify the specific feature, they are designed to detect within the input volume.

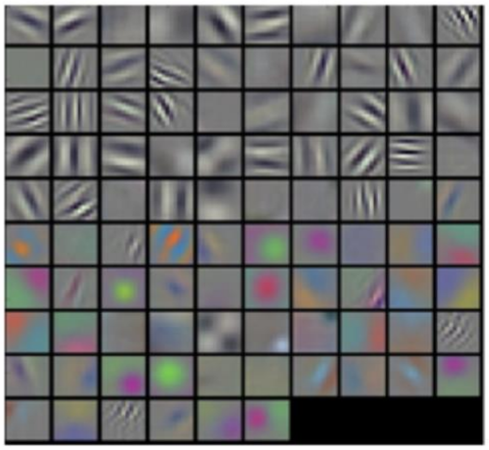
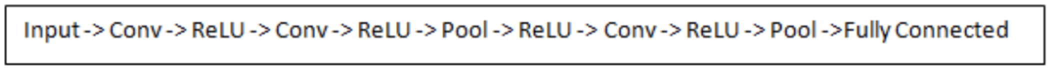


Figure 7. Visualization of first layer conversion filter [Heritage Image Classification by Convolution Neural Networks, Le Van Linh]

2.3.2.5. Deeper in the network

In the conventional architecture of Convolutional Neural Networks, there are additional layers interconnected amid the convolution layers. These layers introduce nonlinearities and facilitate dimension preservation, contributing to the overall robustness of the network. A typical CNN architecture would exhibit the following structure:



Input -> Conv -> ReLU -> Conv -> ReLU -> Pool -> ReLU -> Conv -> ReLU -> Pool ->Fully Connected

Figure 8. Classic CNN architecture [Convolution Neural Networks, Manh-Tu-Vu]

The filters in the convolution layer, which is the initial one, are specifically crafted to detect low-level features such as edges and curves. However, to predict whether an image contains a certain object, the network must be capable of recognizing higher-level features [10]. This necessitates the inclusion of additional layers. In the example above, when another convolution layer is applied, the output of the first convolution layer serves as the input for the second convolution layer. Therefore, each input layer essentially describes the image location where certain low-level features appear. When a set of filters is applied, they generate activations representing high-level features. These features could manifest as semicircles or squares, for instance.

As we traverse through the network and multiple convolution layers, the output consists of activation maps that progressively represent increasingly complex features. It's noteworthy that, with the deepening of the network, the filters start to possess a larger receptive field. This implies that they can take into account information from a broader area compared to the original input volume. Consequently, these filters become more receptive to a larger region of pixel space, enabling them to capture and understand more extensive contextual information in the images being processed.

2.3.2.6. Fully connected layers

This layer integrates information from various parts of the image to make high-level decisions. It takes as input the output of the convolution layer and pooling layer, generating an N-dimensional vector as output, where N corresponds to the number of classes the program needs to choose from. For instance, in a digit classification program with 10 digits, N would be 10. Each element in this N-dimensional vector represents the probability of a particular class. For example, if the vector resulting from a digit classification program is [0.1, 0.1, 0.75, 0, 0, 0, 0, 0, 0, 0.05], it signifies a 10% probability for being 1, a 75% probability for being 2, and a 5% probability for being 9. The functioning of this fully connected layer involves examining the output of the preceding layer and identifying which features are more closely associated and relevant to a specific class.

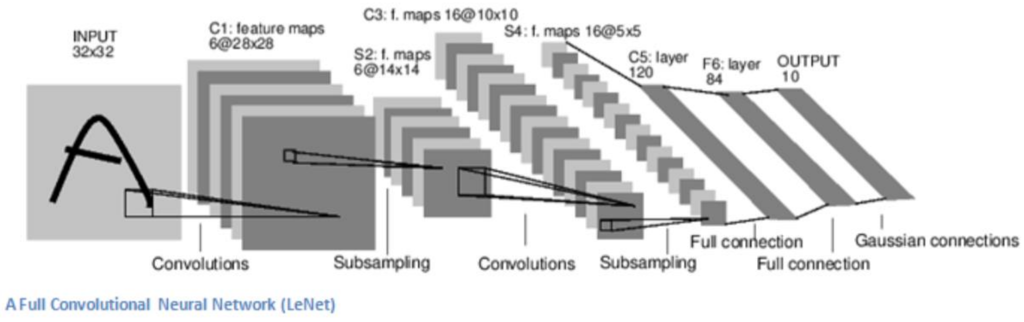


Figure 9. Fully connected CNN scheme [Analytics, Vidhya (2019)]

2.3.2.7. Training (what makes the model work)

Before a Convolutional Neural Network (CNN) begins, the weights and filter values are initially random, lacking knowledge on how to identify specific features like edges, curves, paws, or beaks. Similar to the process of children learning object labels by exposure to different images, the training of CNNs involves a dataset with thousands of images of various objects, each labeled with its corresponding category.

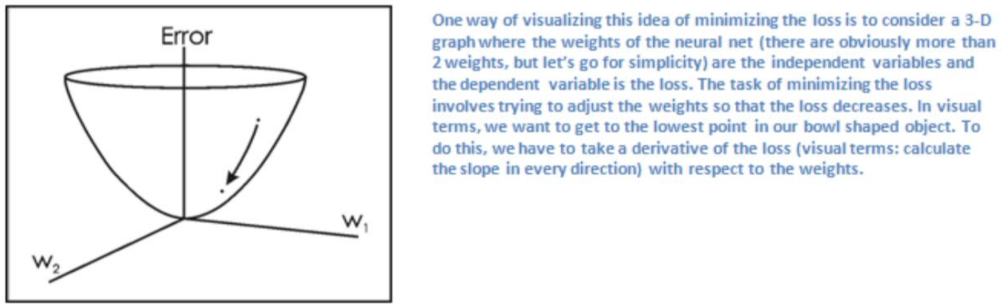
The backpropagation process can be broken down into four distinct stages: forward pass, loss function, back pass, and weight update. During the forward pass, a training image, represented as a 32 x 32 x 3 set of numbers, traverses through the entire network. In the initial training example, with randomly initialized weights and filter values, the output is a result that doesn't prioritize any particular number, resembling something like [.1, .1, .1, .1, .1, .1, .1, .1, .1, .1]. At this stage, the network lacks the capability to recognize low-level features, making it incapable of forming a conclusion about the image classification.

The process moves on to the loss function during backpropagation, where the training data, consisting of an image and its corresponding label, comes into play. For instance, if the label for the first training image is 3, the image label would be represented as [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]. Various methods exist for defining a loss function, with Mean Squared Error (MSE) being the most common. The loss function helps quantify the disparity between the predicted output and the actual label, guiding the network in adjusting its weights and filters during the subsequent stages of backpropagation to enhance its performance.

$$E_{total} = \sum_{\square}^{\square} \square \frac{1}{2}(target - output)^2 \ \Big|_{\square}$$

Figure 10. Loss function

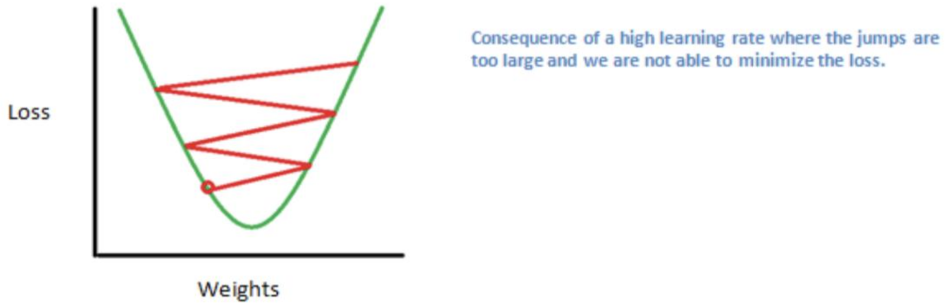
The initial set of training images typically incurs a higher loss. The objective is to ensure that the output of the convolutional network deviates from the training label.



One way of visualizing this idea of minimizing the loss is to consider a 3-D graph where the weights of the neural net (there are obviously more than 2 weights, but let's go for simplicity) are the independent variables and the dependent variable is the loss. The task of minimizing the loss involves trying to adjust the weights so that the loss decreases. In visual terms, we want to get to the lowest point in our bowl shaped object. To do this, we have to take a derivative of the loss (visual terms: calculate the slope in every direction) with respect to the weights.

Figure 11. Minimizing the loss [Mathematics Stacks, 2020]

The objective is to execute a backward pass through the network, identifying which weights contributed the most to the loss, and devising adjustments to minimize the loss. Subsequently, after calculating this derivative, the final step involves updating the weights.



Consequence of a high learning rate where the jumps are too large and we are not able to minimize the loss.

Figure 12. Very high learning rate [Mathematical Stacks, 2020]

The sequence of backpropagation constitutes a training iteration, the aim of which is to optimize the layer weights effectively, ensuring accurate convergence and improved model performance.

How do companies use CNN?

Companies dealing with substantial datasets possess an inherent advantage over their competitors. The abundance of training data allows for a greater number of training iterations, enabling more weight updates and resulting in a better-tuned overall network. For instance, Facebook and Instagram have access to billions of user photos, Pinterest can leverage information from its 50 billion pins, Google can utilize search data, and Amazon can tap into data from the millions of products purchased daily [11]. The scale and diversity of these datasets empower these companies to train and optimize their networks effectively.

CHAPTER 3

THESIS' SPECIFICS

3.1. Analysis constraints

- Constraints as informal text - Informal text serves as constraints during the analysis.
- Operational constraints - Constraints pertaining to operations and functionality.
- Constraints integrated into existing model concepts - The assimilation of constraints into pre-existing model concepts.
- Treating constraints as a distinct concept - Considering constraints as an independent and separate element.
- Constraints implied by the Model Structure - Constraints that are implicitly suggested by the structure of the model.

3.2. Design constraints

- Determining the classes involved
- Determining the objects involved
- Determining the actions involved
- Determining the claim clauses
- Global operations and realization of constraints

3.3. Application constraints

A hierarchical arrangement of relationships may lead to an increased number of classes and a more intricate implementation structure. Therefore, it is recommended to streamline the hierarchical structure into a simpler alternative, such as the conventional flat structure. The latter is characterized by greater decentralization, where decision-making authority is dispersed across the organization. Transforming the hierarchical model into a bipartite, flat structure, comprising classes and straightforward relationships, is relatively straightforward. At the design stage, flat relationships are favored for their simplicity and ease of implementation. Such relationships align with the relational concept found in entity-relationship modeling and are compatible with numerous object-oriented methods.

3.4. Performance requirements

The application manages and interacts with two primary generic components:

- Integrated browser responsible for navigation and Internet access
- Server level: The server houses the core functionalities of the architecture of the paradigm, comprising the following components at this level: a web server, security module, server-side capture engine, preprocessing engine, database system, validation engine, and output module [12].

3.5. Security requirements

- The software constitutes a crucial component of a security system, making its integrity critical. Issues related to its integrity level can be of significant concern.
- If both a system and its program are at a high integrity level, it is essential for the hardware to match that same level of integrity.
- Building flawless code in any language is futile if the hardware and software lack reliability.
- A computer system intended to run high-integrity level software should not concurrently support software of a lower integrity level.
- The highest required integrity level should be enforced across all systems within the same environment.
- Systems with distinct security requirements should be kept separate.

CHAPTER 4

ARCHITECTURE

4.1. Architecture diagram

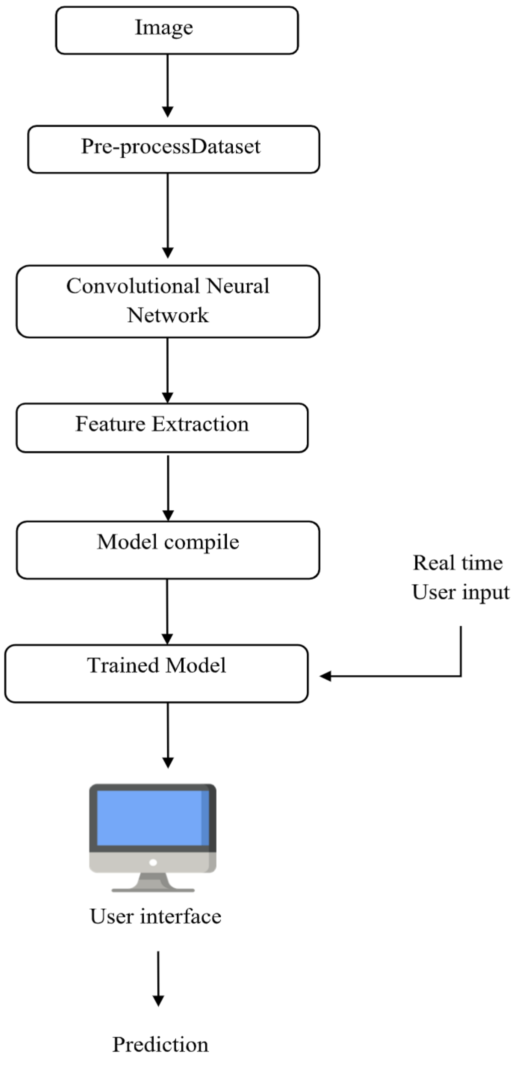


Figure 13. Architecture diagram

The image input refers to the screen that will be used, so the conversion filter.

4.2. Use–Case diagram

The use-case diagram serves to visually represent the functionalities offered by a system concerning the actors, their requirements, and any dependencies among these use cases. It comprises two main components:

- The use case, depicted as a horizontal ellipse, delineates a sequence of actions that yield value to an actor.
- An actor, which can be an individual, organization, or external system, assumes a role in one or more interactions with the system.

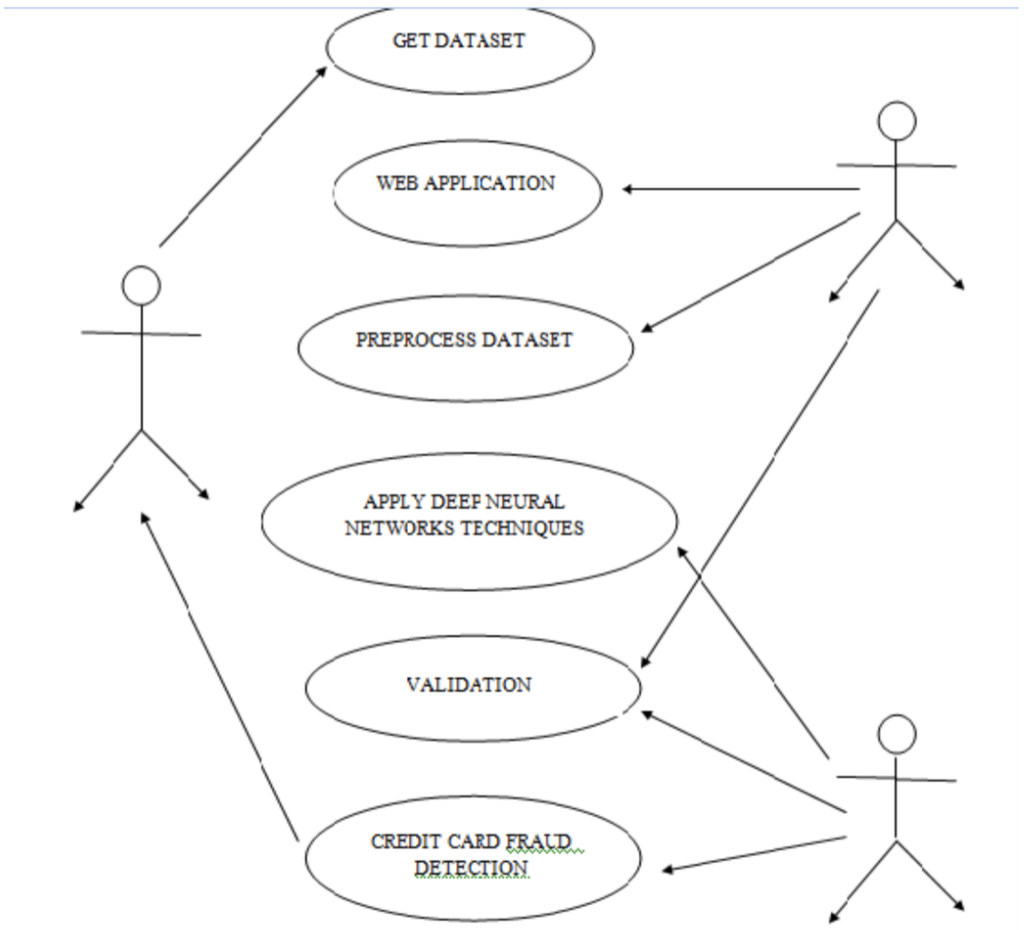

GET DATASET
WEB APPLICATION
PREPROCESS DATASET
APPLY DEEP NEURAL NETWORKS TECHNIQUES
VALIDATION
CREDIT CARD FRAUD DETECTION

Figure 14. Use-Case Diagram

4.3. Activity diagram

The activity diagram provides a visual representation of activity workflows and step-by-step actions, incorporating features like choice, iteration, and concurrency. This diagram offers an overview of the control flow and includes various shapes, such as:

- Rounded rectangles, symbolizing activities.
- Diamond shapes, denoting decision points.
- Bars, indicating the initiation or conclusion of concurrent activities.
- Black circles, signifying the initiation of a workflow.
- Plain circles, representing the ending of a workflow.

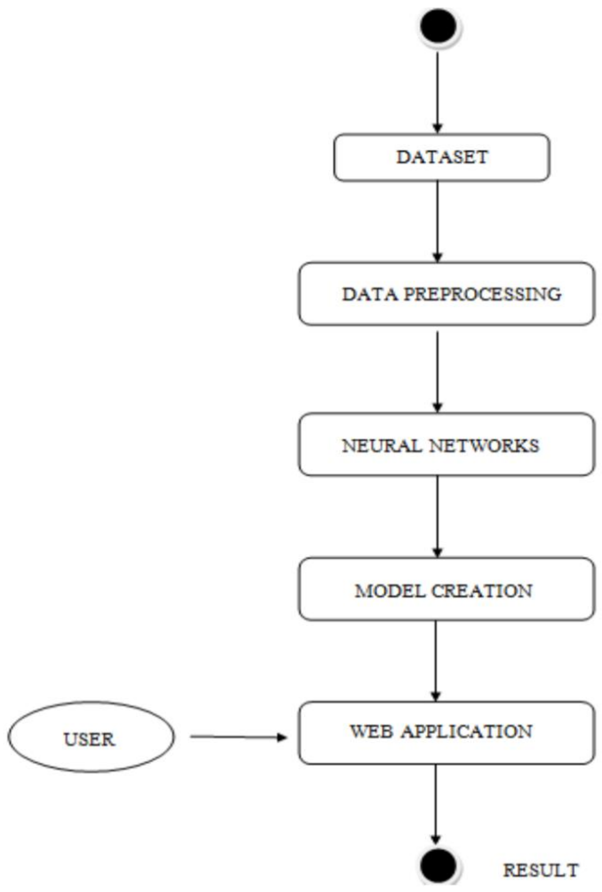


Figure 15. Activity diagram

4.4. Collaboration diagram

A collaboration diagram illustrates the relationships and interactions among objects within a software system. It illustrates the messages that are sent between classes and objects. These diagrams identify each object's functionality and show how a particular use case behaves.

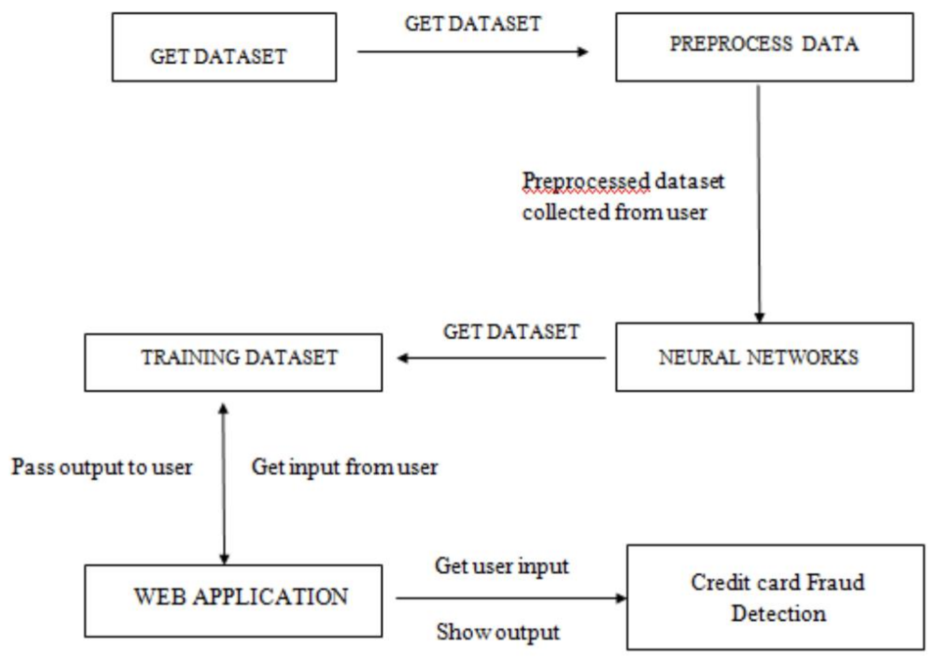


Figure 16. Collaboration diagram

CHAPTER 5

MODULES

5.1. Modules

- Data collection
- Algorithms comparison
- Discoveries

5.2. Comparison among different algorithms

In my analysis I considered and evaluated several classifiers in order to determine their effectiveness in detecting fraudulent transactions. The models that I considered included:

- Random forest
- Gradient Boosting
- K-nearest neighbor
- Support Vector Classifier
- Logistic Regression

Random Forest and Gradient Boosting are often preferred for classification tasks, such as credit card detection, as a result of their adaptive ability to different datasets.

Logistic Regression is a common choice for simple binary classification tasks.

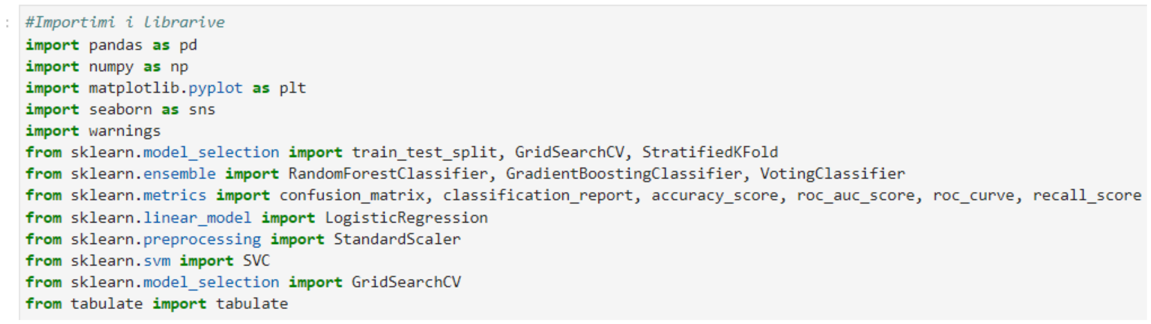
K-Nearest Neighbors (KNN) is useful for a simple approach and does not require model training, however it poses difficulty for new data arrivals and performs poorly on large datasets.

Support Vector Classifier (SVC) is used on datasets with a distinct separation between classes, but it needs good calibration on hyper parameters.

Convolutional Neural Network (CNN) are mainly used for the analysis of complex data structures and they are more suitable for datasets that contain images or structured information.

5.2.1 Followed steps for comparing the algorithms

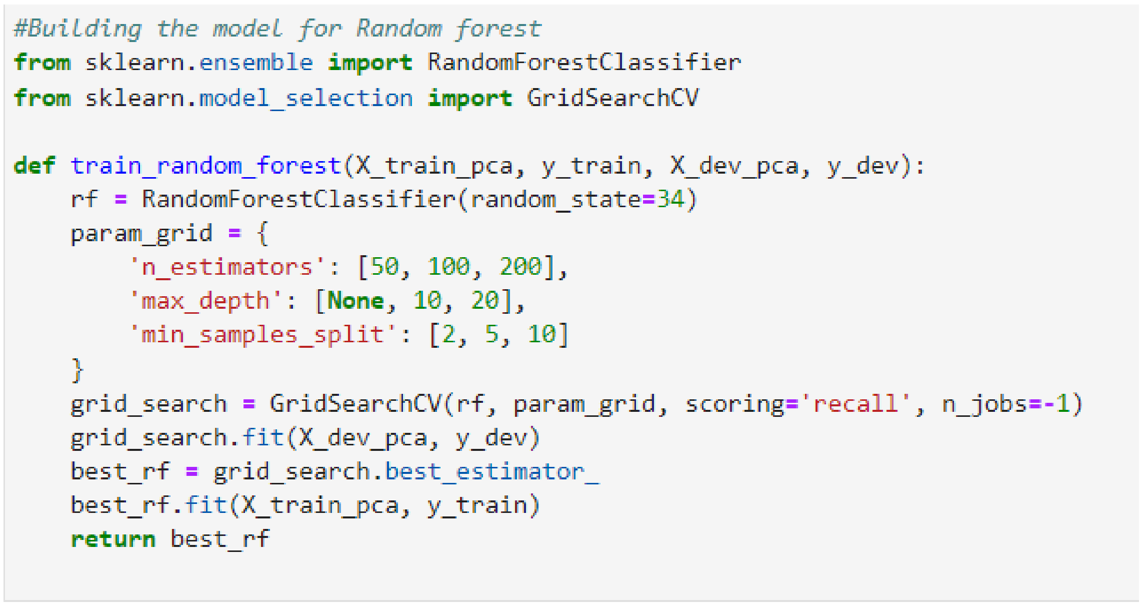
1. Initially I imported the libraries.



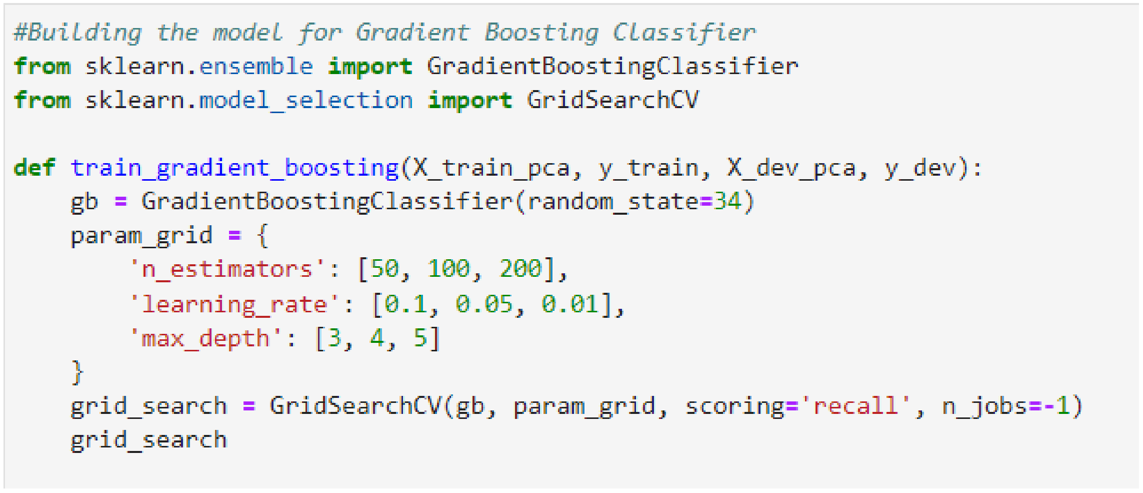
```
#Importimi i Librarive
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, VotingClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, roc_auc_score, roc_curve, recall_score
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from tabulate import tabulate
```

Figure 17. Libraries importing in Jupyter for comparison between models

2. Then I built the models for each of the algorithms

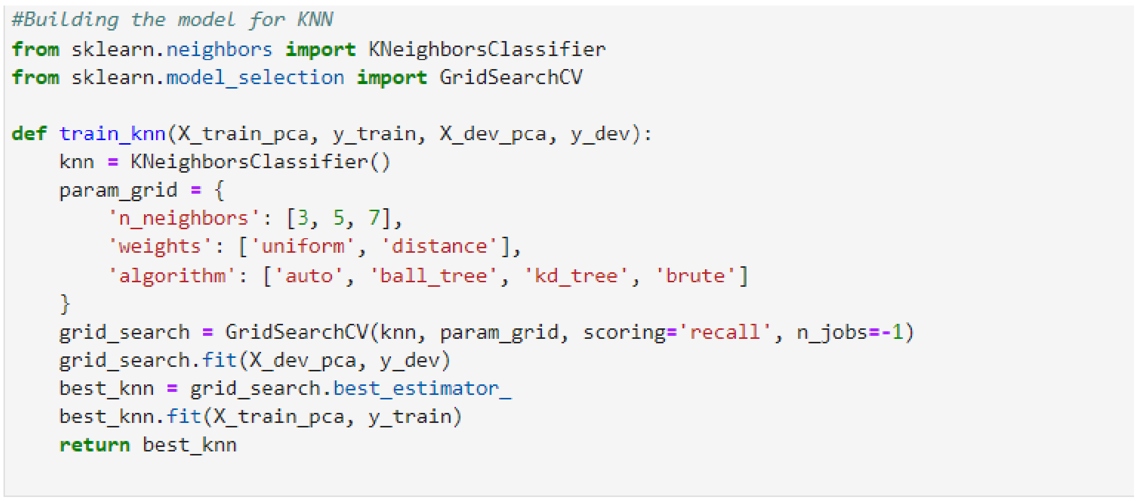


```
#Building the model for Random forest
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

def train_random_forest(X_train_pca, y_train, X_dev_pca, y_dev):
    rf = RandomForestClassifier(random_state=34)
    param_grid = {
        'n_estimators': [50, 100, 200],
        'max_depth': [None, 10, 20],
        'min_samples_split': [2, 5, 10]
    }
    grid_search = GridSearchCV(rf, param_grid, scoring='recall', n_jobs=-1)
    grid_search.fit(X_dev_pca, y_dev)
    best_rf = grid_search.best_estimator_
    best_rf.fit(X_train_pca, y_train)
    return best_rf
```



```
#Building the model for Gradient Boosting Classifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV

def train_gradient_boosting(X_train_pca, y_train, X_dev_pca, y_dev):
    gb = GradientBoostingClassifier(random_state=34)
    param_grid = {
        'n_estimators': [50, 100, 200],
        'learning_rate': [0.1, 0.05, 0.01],
        'max_depth': [3, 4, 5]
    }
    grid_search = GridSearchCV(gb, param_grid, scoring='recall', n_jobs=-1)
    grid_search
```

Figure 18. Building the model for Random Forest and Gradient Boosting


```python
#Building the model for KNN
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV

def train_knn(X_train_pca, y_train, X_dev_pca, y_dev):
    knn = KNeighborsClassifier()
    param_grid = {
        'n_neighbors': [3, 5, 7],
        'weights': ['uniform', 'distance'],
        'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']
    }
    grid_search = GridSearchCV(knn, param_grid, scoring='recall', n_jobs=-1)
    grid_search.fit(X_dev_pca, y_dev)
    best_knn = grid_search.best_estimator_
    best_knn.fit(X_train_pca, y_train)
    return best_knn
```

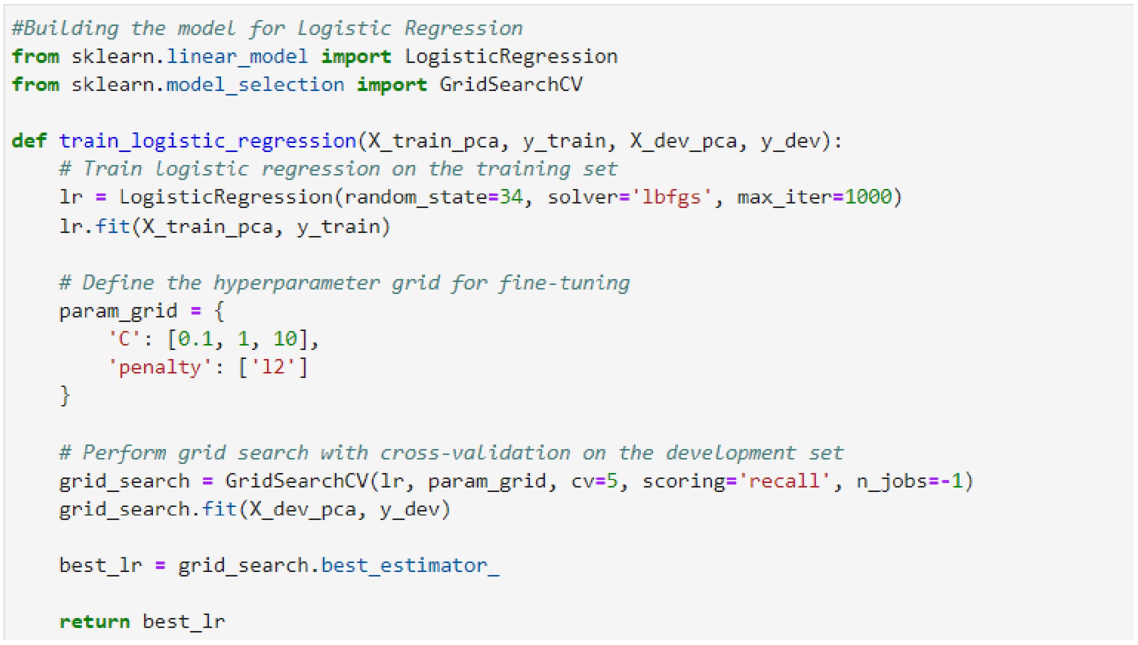
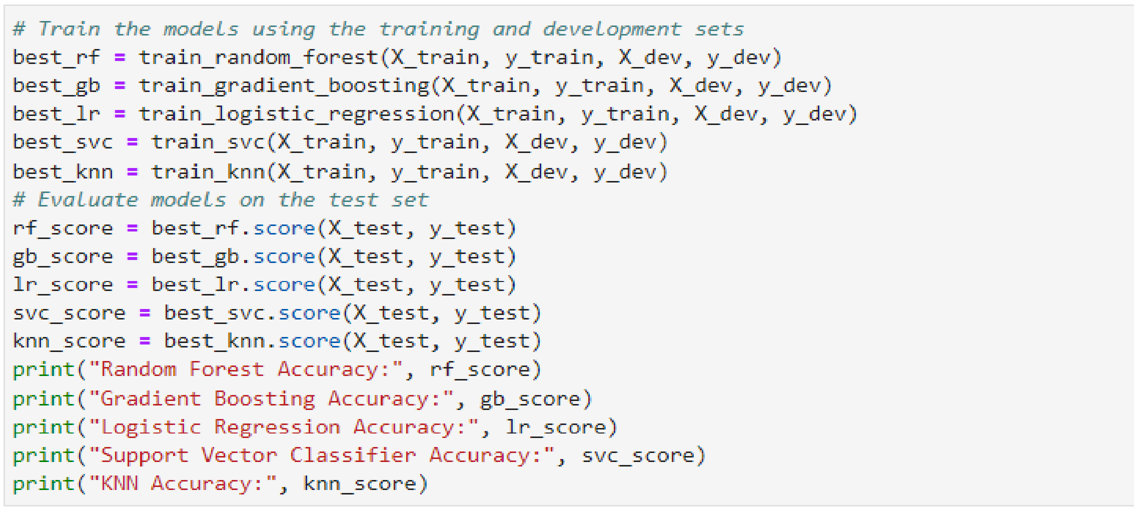

```python
#Building the model for Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

def train_logistic_regression(X_train_pca, y_train, X_dev_pca, y_dev):
    # Train logistic regression on the training set
    lr = LogisticRegression(random_state=34, solver='lbfgs', max_iter=1000)
    lr.fit(X_train_pca, y_train)

    # Define the hyperparameter grid for fine-tuning
    param_grid = {
        'C': [0.1, 1, 10],
        'penalty': ['l2']
    }

    # Perform grid search with cross-validation on the development set
    grid_search = GridSearchCV(lr, param_grid, cv=5, scoring='recall', n_jobs=-1)
    grid_search.fit(X_dev_pca, y_dev)

    best_lr = grid_search.best_estimator_

    return best_lr
```

Figure 19. Building the model for KNN and Logistic Regression

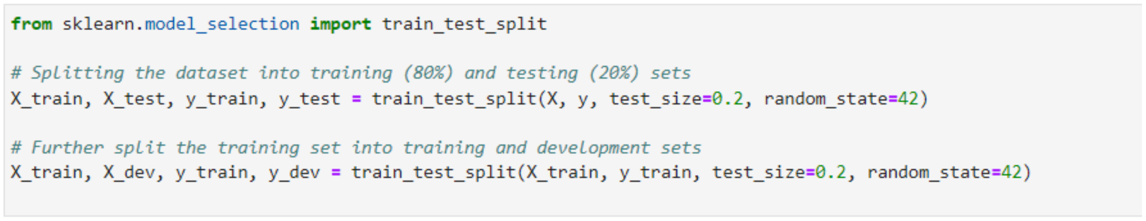
3. Then I trained the models.



```
# Train the models using the training and development sets
best_rf = train_random_forest(X_train, y_train, X_dev, y_dev)
best_gb = train_gradient_boosting(X_train, y_train, X_dev, y_dev)
best_lr = train_logistic_regression(X_train, y_train, X_dev, y_dev)
best_svc = train_svc(X_train, y_train, X_dev, y_dev)
best_knn = train_knn(X_train, y_train, X_dev, y_dev)
# Evaluate models on the test set
rf_score = best_rf.score(X_test, y_test)
gb_score = best_gb.score(X_test, y_test)
lr_score = best_lr.score(X_test, y_test)
svc_score = best_svc.score(X_test, y_test)
knn_score = best_knn.score(X_test, y_test)
print("Random Forest Accuracy:", rf_score)
print("Gradient Boosting Accuracy:", gb_score)
print("Logistic Regression Accuracy:", lr_score)
print("Support Vector Classifier Accuracy:", svc_score)
print("KNN Accuracy:", knn_score)
```

Figure 20. Training the models with training sets

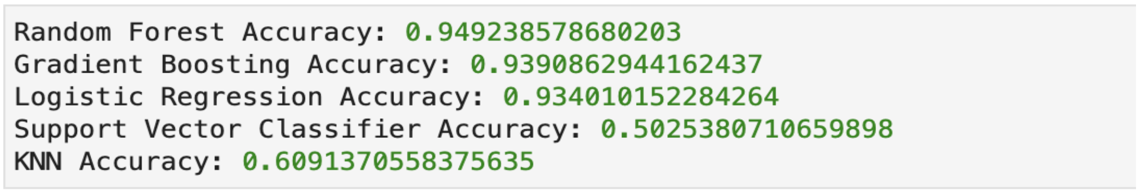
4. I split the dataset into training and testing sets.



```
from sklearn.model_selection import train_test_split

# Splitting the dataset into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Further split the training set into training and development sets
X_train, X_dev, y_train, y_dev = train_test_split(X_train, y_train, test_size=0.2, random_state=42)
```

Figure 21. Dividing the dataset into training and testing sets

5. The accuracy of the models' results was then calculated and displayed.



```
Random Forest Accuracy: 0.949238578680203
Gradient Boosting Accuracy: 0.9390862944162437
Logistic Regression Accuracy: 0.934010152284264
Support Vector Classifier Accuracy: 0.5025380710659898
KNN Accuracy: 0.6091370558375635
```

Figure 22. Displaying the models' accuracy

5.2.2 Comparison’s result

Each algorithm in general has its own advantage and disadvantage, and their performance varies on the nature of data and what is aimed to be studied with them. However, given that with CNN, an accuracy of 98% was attained through different training epochs, it’s worth stating that the CNN algorithm that I decided to work with, is the most suitable and effective one in the context of fraudulent transaction detection.

5.3 Collecting the dataset

The CNN model's deep learning (DL) algorithm underwent modifications through the addition of extra layers for feature extraction and the classification of credit card transactions as either fraudulent or not. The primary objective of this model is to detect fraudulent transactions utilizing deep learning algorithms. In the initial phase, the unbalanced CCF numeric dataset is converted into a balanced dataset by eliminating non-fraudulent transactions. In real-world transactions, the imbalance between fraudulent and normal classes is inherent to the nature of the problem. The proposed model employs a convolutional neural network with a 14-layer architecture on the balanced dataset to validate its effectiveness. This architecture achieved training and validation accuracies of 94.6% and 95.8%, respectively.

Distinctively from traditional neural networks that rely on multiplications of matrixes, convolutional networks employ a Convolution technique. In mathematical terms, this method is an operation on two functions, producing a third one that demonstrates how the shape of one function is modified by the other.

5.4 Layers in the CNN

In the CNN model, six distinct layers have been defined: the input layer, Convolutional layer (Convo layer), Pooling layer, Fully Connected layer, SoftMax (logistic layer), and the output layer. The input layer comprises of image data, represented by three-dimensional matrices that are transformed into a single column. The Convolutional layer serves as the feature extraction layer, and the Pooling layer diminishes the spatial volume of the input image, making it computationally efficient. The subsequent layer is the Fully Connected layer, involving weights, biases, and neurons to establish connections between different layers' neurons. Through training, this layer classifies photos into multiple categories. Finally, the logistic layer is utilized for binary classification, assigning a label of 0 to the normal class and 1 to the abnormal class.

Given the typical imbalance between fraudulent and normal classes, the initial step involves transforming the unbalanced CCF data into a balanced dataset by excluding non-fraudulent transactions. The proposed model comprises 14 layers: a convolutional layer with a 32 x 2 kernel size and a ReLU activation function, followed by a group normalization layer and a dropout layer with a dropout rate of 0.2. Another convolutional layer follows with a 64 x 2 kernel size and a ReLU activation function, accompanied by a batch normalization layer and a dropout layer with a 0.5 dropout rate. This sequence is repeated with an additional layer where only the dropout layer changes to 0.25. Three more dense layers are included, with the first having an activation function of 100, the second with 50, and the third with a ReLU of 25. Finally, a dense layer for classification with a sigmoid activation function is appended. After 100 iterations, the accuracy exceeds 90%.

CHAPTER 6

CODING

6.1. Coding

After finalizing the design aspect of the system, the next steps involve coding and testing. The coding phase transforms the conceptual system design into operational reality by translating it into code using a specific programming language. For this reason, an efficient coding style is required that whenever new changes are made, it easily adapts them to the system.

6.2. Coding standards

Coding standards consist of programming guidelines that prioritize the visual structure and presentation of the code. They are designed to enhance code readability, comprehension, and maintainability. During the coding phase, the developed plan from the design phase is implemented, and adherence to coding standards becomes crucial. The specifications of the coding should enable any programmer to comprehend it easily and make necessary modifications when required. Some essential standards to achieve the aforementioned objectives include:

1. Simplicity and clarity
2. Naming conventions
3. Value conventions
4. Writing and commenting standard
5. Message box format
6. Exceptions and error handling

6.2.1. Writing and commenting standard

Scriptwriting is a process in which indentation is highly important in order to make the code readable and easy to understand. Conditional and loop statements should be matched properly to facilitate understanding. Comments are also included to better know what to expect when running the code.

6.2.2. Message box format

When requesting information from the user, it is essential that the user can easily understand the messages presented to them. To achieve this, a specific format has been established for displaying messages, as outlined below:

- X – The user has performed an illicit operation
- ! – Denotes messages containing important information for the user

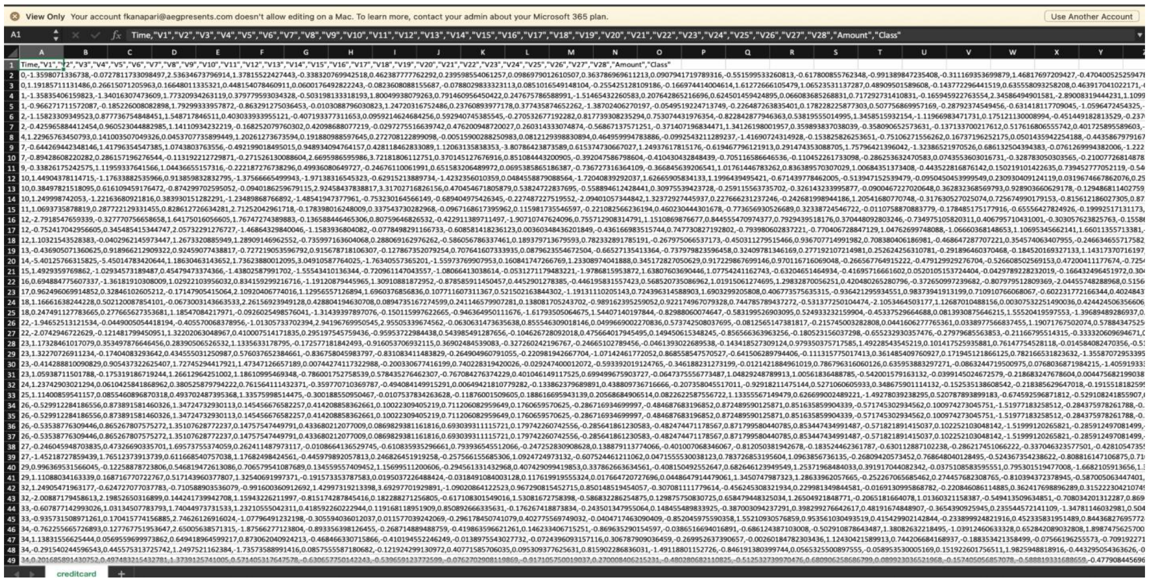
CHAPTER 7

DATA AND CODE PREPARATION

7.1. Dataset description

The dataset comprises transactions made by European citizens using credit cards over a span of two days, with a total of 284,807 transactions, including 492 instances of fraud. Notably, the data exhibits significant imbalance, as the positive class (fraudulent transactions) constitutes only 0.172% of the entire dataset. Key characteristics of the dataset include:

- Only numeric data variables resulting from Principal Component Analysis (PCA) transformation, denoted as columns V1, V2, … V28. PCA transformation was applied to reduce the dimensionality of the numeric dataset while still preserving its information.
- Due to confidentiality constraints, the original features of the data are undisclosed.
- Attributes that include "time" (representing seconds elapsed since the first transaction) and "value" (sum of transactions, indicating cost dependency).
- The "class" attribute functions as the dependent variable, assuming a value of 1 for fraud and 0 for normal transactions.



View Only Your account fkanapari@aegpresents.com doesn't allow editing on a Mac. To learn more, contact your admin about your Microsoft 365 plan. Use Another Account

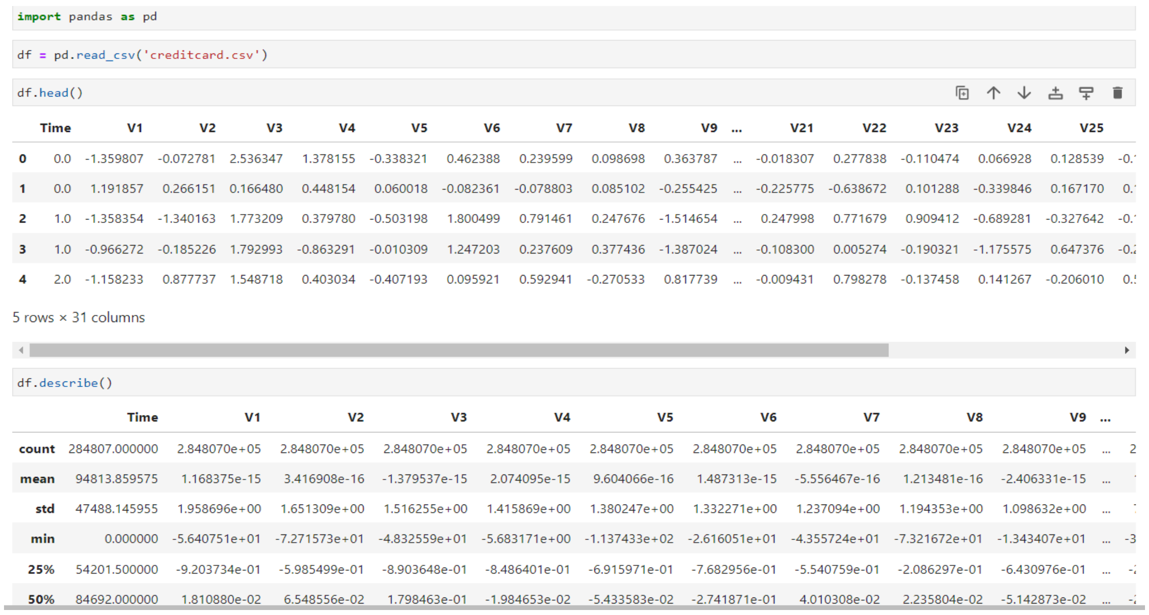
A1 Time,"V1","V2","V3","V4","V5","V6","V7","V8","V9","V10","V11","V12","V13","V14","V15","V16","V17","V18","V19","V20","V21","V22","V23","V24","V25","V26","V27","V28","Amount","Class"

```
1  Time,"V1","V2","V3","V4","V5","V6","V7","V8","V9","V10","V11","V12","V13","V14","V15","V16","V17","V18","V19","V20","V21","V22","V23","V24","V25","V26","V27","V28","Amount","Class"
2  0,-1.3598071336738,-0.0727811733098497,2.53634673796914,1.37815522427443,-0.338320769942518,0.462387777762292,0.239598554061257,0.0986979012610507,0.363786969611213,0.0907941719789316,-0.551599533260813,-0.617800855762348,-0.991389847235408,-0.311169353699879,1.46817697209427,-0.470400525259478
3  0,1.19185711131486,0.26615071205963,0.16648011335321,0.448154078460911,0.0600176492822243,-0.0823608088155687,-0.0788029833323113,0.0851016549148104,-0.255425128109186,-0.166974414004614,1.61272666105479,1.06523531137287,0.48909501589608,-0.143772296441519,0.635558093258208,0.463917041022171,-
4  1,-1.35835406159823,-1.34016307473609,1.77320934263119,0.379779593034328,-0.503198133318193,1.80049938079263,0.791460956450422,0.247675786588991,-1.51465432260583,0.207642865216696,0.624501459424895,0.066083685268831,0.717292731410831,-0.165945922763554,2.34586494901581,-2.89008319444231,1.1099
5  1,-0.966271711572087,-0.185226008082898,1.79299333957872,-0.863291275036453,-0.0103088796030823,1.24720316752486,0.23760893977178,0.377435874652262,-1.38702406270197,-0.0549519224713749,-0.226487263835401,0.178228225877303,0.507756869957169,-0.28792374549456,-0.631418117709045,-1.0596472454325,-
6  2,-1.15823309349523,0.877736754848451,1.548717846511,0.403033933955121,-0.407193377311653,0.0959214624684256,0.592940745385545,-0.270532677192282,0.817739308235294,0.753074431976354,-0.822842877946363,0.53819555014995,1.3458515932154,-1.11966983471731,0.175121130008994,-0.451449182813529,-0.237
7  2,-0.425965884412454,0.960523044882985,1.14110934232219,-0.168252079760302,0.42098688077219,-0.0297275516639742,0.476200948720027,0.260314333074874,-0.56867137571251,-0.371407196834471,1.34126198001957,0.359893837038039,-0.358090652573631,-0.137133700217612,0.517616806555742,0.401725895589603,-
8  4,1.22965763450793,0.141003507049326,0.0453707735899449,1.20261273673594,0.191880988597645,0.272708122899098,-0.00515900288250983,0.0812129398830894,0.464959994783886,-0.0992543211289237,-1.41690724314928,-0.153825826253651,-0.75106271556262,0.16737196252175,0.0501435942254188,-0.4435867979167
9  7,-0.644269442348146,1.41796354547385,1.0743803763556,-0.492199018495015,0.948934094764157,0.428118462833089,1.12063135838353,-3.80786423873589,0.615374730667027,1.24937617815176,-0.619467796121913,0.291474353088705,1.75796421396042,-1.32386521970526,0.686132504394383,-0.0761269994382006,-1.222
10 7,-0.89428608220282,0.286157196276544,-0.113192212729871,-0.271526130088604,2.6695986595986,3.72181806112751,0.370145127676916,0.851084443200905,-0.392047586798604,-0.410430432848439,-0.705116586646536,-0.110452261733098,-0.286253632470583,0.0743553603016731,-0.328783050303565,-0.21007726814878
11 9,-0.33826175242575,1.11959337641566,1.04436655157316,-0.222187276738296,0.49936080649727,-0.24676110061991,0.651583206489972,0.0695385865186387,-0.736727316364109,-0.366845639206541,1.01761446783262,0.836389570307029,1.00684351373408,-0.443522816876142,0.150219101422635,0.739452777052119,-0.54
12 10,1.44904378114715,-1.17633882535966,0.913859832832795,-1.37566665499943,-1.97138316545323,-0.62915213889734,-1.4232356010359,0.0484558879088564,-1.72040839292037,1.62665905834133,1.1996439495421,-0.671439778462005,-0.513947152539479,-0.0950450453999549,0.230930409124119,0.0319674667862076,0.25
13 10,0.38497821518095,0.616109459176472,-0.874299702595052,-0.0940186259679115,2.92458437838817,3.31702716826156,0.470454671805879,0.53824722837695,-0.558894612428441,0.30975539423728,-0.259115563735702,-0.326143233995877,-0.0900467227020648,0.362832368569793,0.928903660629178,-0.129486811402759,
14 10,1.249998742053,-1.22163680921816,0.383930151282291,-1.23489868766892,-1.48541947377961,-0.753230164566149,-0.689404975426345,-0.227487227519552,-2.09401057344842,1.32372927445937,0.227666231237246,-0.242681998944186,1.20541680770748,-0.317630527025074,0.725674990179153,-0.815612186027305,0.87
15 11,1.0693735878819,0.287722129331455,0.828612726634281,2.71252042961718,-0.178398016248009,0.337543730282968,-0.0967168617395962,0.115981735546597,-0.221082566236194,0.460230444301678,-0.773656930526689,0.32338724546722,-0.0110758870883779,-0.178485175177916,-0.65556427824926,-0.19992517131173,
16 12,-2.7918547659339,-0.327770756658658,1.64175016056605,1.76747274389883,-0.136588446465306,0.80759646826532,-0.422911389711497,-1.90710747624096,0.755712908314791,1.1510869876677,0.844555470974377,0.7929439518176,0.370448092803246,-0.734975105820311,0.406795710431001,-0.303057623825763,-0.1558
17 12,-0.752417042956605,0.345485415344747,2.05732291276727,-1.46864329840046,-1.1583936804082,-0.0778498291166733,-0.608581418236123,0.00360348436201849,-0.436166983515744,0.747730827192802,-0.793980602837221,-0.770406728847129,1.04762699748088,-1.06660368148653,1.10695345662141,1.66011355713381,-
18 12,1.10321543528383,-0.0402962145973447,1.2673320885949,1.28909146962552,-0.735997163604068,0.288069162976262,-0.586056786337461,0.189379713679593,0.782332891785191,-0.267975066537173,-0.450311279515466,0.936707714991982,0.708380406186981,-0.468647287707221,0.354574063407955,-0.246634655717582,
19 13,-0.436905071360625,0.918966212909322,0.92459077438817,-0.727219053596792,0.915678718106307,-0.127867352079254,0.707641607333935,0.0879623554672504,-0.66527135413364,-0.737979823596458,0.32409781346169,0.277192107214981,0.252624256310781,-0.291896460370468,-0.184520169327133,1.14317370716197,
20 14,-5.40125766315825,-5.45014783420644,1.18630463143652,1.73623880012095,3.04910587764025,-1.76340557365201,-1.55973769907953,0.160841747266769,1.23308974041888,0.345172827050629,0.917229867699146,0.970116716069048,-0.266567764915222,-0.479129929276704,-0.526608502569153,0.47200411177674,-0.7254
21 15,1.4929359769862,-1.02934573189487,0.45479473374366,-1.43802587991702,-1.55543410136344,-0.720961147043557,-1.08066413038614,-0.0531271179483221,-1.9786815953872,1.63807603690446,1.07754241162743,-0.63204651464934,-0.41695716661602,0.0520105153724404,-0.0429789228232019,-0.166432496451972,0.304
22 16,0.694884775607337,-1.36181910308009,1.02922103956032,0.834159299216716,-1.19120879445965,1.30910881872952,-0.878585911450457,0.4452901278385,-0.446195831557423,0.568520735086962,1.01915061274695,1.29832870056251,0.420480265280796,-0.372650997239682,-0.807979512809369,-2.04455748288968,0.5156
23 17,0.962496069914852,0.32846102605212,-0.17147905415064,2.10920406774016,1.12956557126894,1.6960376856836,0.107711607311367,0.521502163844302,-1.19131110205143,0.724396314588903,1.6903299205808,0.406773575635315,-0.936421295934551,0.98373941913199,0.710910766068067,-0.602231772166344,0.4024843
24 18,1.16661638244228,0.502120087854101,-0.0673003143663533,2.26156923949128,0.428804194630708,0.0894735167274599,0.241146579907281,0.138081705243702,-0.989162395259052,0.922174967079328,0.744785789437272,-0.531377250104474,-2.1053464503177,1.12687010488156,0.00307532251490036,0.424424506356606,
25 18,0.247491127783665,0.277665627353681,1.18547084217971,-0.0926025498576041,-1.31439397897076,-0.150115997622665,-0.946364950111676,-1.61793505064675,1.54407140197844,-0.82988060074647,-0.583199526903095,0.524933232159904,-0.45337529664688,0.0813930875646215,1.55520419597553,-1.39689489286937,0.
26 22,-1.94652513121534,-0.0449005054418194,-0.405570068378956,-1.01305733702394,2.94196769950545,2.95505339674562,-0.0630631473635638,0.855546309018146,0.0499669002270836,0.573742508037695,-0.0812565147381817,-0.215745003282808,0.0441606277765361,0.0338977566837455,1.19071767502074,0.57884347525
27 22,-2.0742946722629,-0.121481799450951,1.32202063048967,0.410007514171835,0.295197545759436,-0.95953722984438,0.543985491287656,-0.104626728092018,0.475664017945495,0.149450615348245,-0.856566363963256,-0.180523156037298,-0.655232930357476,-0.27979685563853,-0.21166795514315,-0.333320609694671,0
28 23,1.17328461017079,0.353497876646456,0.28390506526532,1.1335633178795,-0.172577181842493,-0.916053706932115,0.3690248453908 3,-0.327260242196767,-0.24665102789456,-0.0461393022689538,-0.143418527309124,0.979350375717585,1.49228543545219,0.101417525935881,0.76147754528118,-0.014584082470356,-0.51
29 23,1.32270726911234,-0.17404083293642,0.434555031250987,0.576037652384661,-0.836758045983797,-0.831083411483829,-0.264904960791055,-0.220981942667704,-1.07142461772052,0.868558547570527,-0.641506289794406,-0.111315775017413,0.361485409760927,0.171945121866125,0.782166531823632,-1.35587072953395
30 23,-0.414288810090829,0.905437322625407,1.72745294417921,1.47347126657189,0.00744274117322988,-0.200330677416199,0.740228319420026,-0.029247400012072,-0.593392019124765,-0.346188231273199,-0.0121421884961019,0.786796316060126,0.635953883297271,-0.0863244719500975,0.0768036871984215,-1.40591933
31 23,1.05938711501788,-0.175319186719244,1.26612964251002,1.18610995469348,-0.786001752758539,0.578435276462307,-0.767084276374229,0.401046149117525,0.699499675903727,-0.0647375556773487,1.04829248789913,1.00561836488785,-0.542001579163132,-0.0399145024672579,-0.218683247678604,0.004475682199038
32 24,1.23742903021294,0.0610425841868962,0.380525879794222,0.761564111432371,-0.359770710369787,-0.494084149915291,0.0064942181077928 2,-0.133862379689891,0.438809736716666,-0.207358045517011,-0.92918211475144,0.5271060605933,0.348675901114132,-0.152535138608542,-0.218385629647018,-0.1915518182595
33 25,1.11400859541157,0.0855460896870318,0.493702487395368,1.33575998514475,-0.300188550950467,-0.0107537834263628,-0.11876001509605,0.188616695943139,0.20568684906514,0.0822622587556722,1.13355567149479,0.626699002489221,-1.49278039238295,0.520787893899183,-0.674592596871812,-0.529108241855907,
34 26,-0.529912284186556,0.873891581460326,1.34724732930113,0.145456676582257,0.414208858362661,0.10022309405219,0.711206082959649,0.1760659570625,-0.286716934699997,-0.484687683196852,0.872489590125871,0.851635859904339,-0.571745302934562,0.100974273045751,-1.51977183258512,-0.284375978261788,-0.
35 26,-0.529912284186556,0.873891581460326,1.34724732930113,0.145456676582257,0.414208858362661,0.10022309405219,0.711206082959649,0.1760659570625,-0.286716934699997,-0.484687683196852,0.872489590125871,0.851635859904339,-0.571745302934562,0.100974273045751,-1.51977183258512,-0.284375978261788,-0.
36 26,-0.53538776309446,0.865267807575272,1.35107628772237,0.14757547449791,0.433680212077009,0.0869829381161816,0.693039311115721,0.179742260742556,-0.285641861230583,-0.482474471178567,0.871799580440785,0.853447434991487,-0.571821891415037,0.102252103048142,-1.51999120265821,-0.285912497081499,-
37 26,-0.53538776309446,0.865267807575272,1.35107628772237,0.14757547449791,0.433680212077009,0.0869829381161816,0.693039311115721,0.179742260742556,-0.285641861230583,-0.482474471178567,0.871799580440785,0.853447434991487,-0.571821891415037,0.102252103048142,-1.51999120265821,-0.285912497081499,-
38 27,-0.246045948703835,0.47326690335701,1.69573755374059,0.262411487973117,-0.0108664136529745,-0.610835935296661,0.793936545512066,-0.247252830908628,0.138879113774066,-0.401007068346067,-0.812050381942678,-0.183524462361787,-0.630112887102238,-0.286217451066222,-0.337046323577501,-0.42810547355
39 27,-1.45218727859439,1.76512373913739,0.611668540757038,1.17682498424561,-0.445979892057813,0.246826451919258,-0.257566155685306,1.0924724973132,-0.607524461211062,0.0471555530038123,0.783726853195604,1.0963856736135,-0.26809420573452,0.768648040128495,-0.524367354238622,-0.808816147106875,0.71
40 29,0.996369531566045,-0.12258878723806,0.546819472613086,0.706579541087689,0.134559557409452,1.15699511200606,-0.294561331432968,0.407429099419853,0.337862663634561,-0.408150492552647,0.682646123949549,1.25371968484033,0.391917044082342,-0.0375108583595551,0.795301519477008,-1.6682105913656,1.3
41 29,1.11088034163339,0.168716770722767,0.517143960377807,1.32540691997371,-0.191573353787583,0.0195037226488424,-0.0318491084003128,0.117619919555324,0.017664720727696,0.0448647914479061,1.3450747987323,1.28633962057665,-0.252267065685462,0.274457682308765,-0.810394372378945,-0.587005063447401,
42 32,1.24905471963177,-0.624727077037783,-0.710588903536079,-0.991600360912692,1.42997319213398,3.69297701929891,-1.09020864122523,0.967290815452715,0.850148519454057,-0.307081111779614,-0.456245308321934,0.229981349844581,-0.0169130995868782,-0.220846086114885,0.362417698896289,0.315222304210749
43 32,-2.00887179458613,2.19852650316899,0.144241739942708,1.15943226211997,-0.815174287845416,0.182288271256805,-0.617108301549016,1.53081672758398,-0.586832286254875,0.129875750830725,0.658479448325034,1.26504921848771,-0.2065181664078,1.01360321158387,-0.549413509634851,-0.708034201312287,0.869
44 33,-0.607877142993026,1.03134507783793,1.74044973731533,1.23210555042311,0.418592260222944,0.119168118951909,0.850892666335631,-0.176267418873834,-0.243501347955064,0.148455489833925,-0.387003094237291,0.398299276642617,0.48191674848907,-0.3654390925945,0.235544572141109,-1.34781146032981,0.504
45 33,-0.935731508971261,0.170415774156885,2.74626126916024,-1.07796491232198,-0.305594036012037,0.011577039242069,-0.296178450741079,0.402775569749032,-0.0404717463090409,-0.852045975590358,1.55210930576859,0.953561030493519,0.415429902142844,-0.233899924821916,0.452335831951489,0.84436827695772
46 34,-0.762255665726893,0.127767751953647,2.65005638571315,-1.87566277123804,-0.893356398126455,-0.268714889488759,-0.419863596621261,0.146233406715251,-0.869635290154597,-0.0386516694016891,-0.686124387103008,-0.502910878643487,1.38082632218495,-1.03912460633328,0.652842089032808,1.8987475625700
47 34,1.13831556625444,0.0569559699973862,0.649418964599217,0.873062040924213,-0.468466330715866,-0.410194552246249,-0.0138975543027732,-0.0724396093157116,0.306787909036459,-0.269952637390657,-0.00260184782303436,1.12430421589913,0.744206684168937,-0.188353421358499,-0.07566196255573,-0.709192271
48 34,-0.291540244596543,0.445575313725742,1.2497521162384,-1.73573588991416,0.0857555587180682,-0.121924299130972,0.40771585706035,0.095309377625631,0.815902286836031,-1.49118801152726,-0.846191380399744,0.0565325500897555,-0.058953530005169,0.151922601756511,1.9825948818916,-0.443295054363626,-0
49 34,0.201685891430752,0.497483215432781,1.3739125741005,0.571405317647578,-0.630657750142243,-0.539659123772599,-0.0762702908119869,-0.917105750019037,0.270008406215231,-0.480280682110825,-0.512532739970476,0.680906258686799,0.089923036521968,-0.157405056857078,-0.588819331688659,-0.477908445696
```

creditcard

Figure 23. The dataset used for the model [Kaggle, 2013]

In order to explore this dataset, a PCA transformation has been conducted in order to reduce its large dimensions into a more understandable one.



```python
import pandas as pd
```

```python
df = pd.read_csv('creditcard.csv')
```

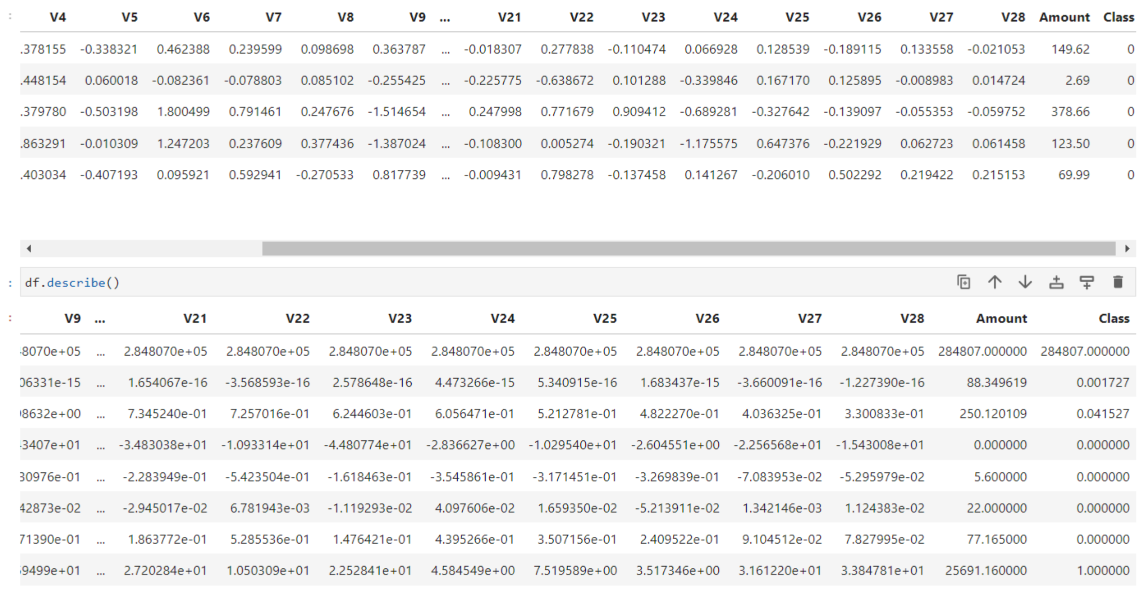
```python
df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.

5 rows × 31 columns

```python
df.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...	2
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-16	-2.406331e-15	...	
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	...	
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	...	-3
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	...	-
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	...	-



V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

```
df.describe()
```

V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
8070e+05	...	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	284807.000000	284807.000000
06331e-15	...	1.654067e-16	-3.568593e-16	2.578648e-16	4.473266e-15	5.340915e-16	1.683437e-15	-3.660091e-16	-1.227390e-16	88.349619	0.001727
8632e+00	...	7.345240e-01	7.257016e-01	6.244603e-01	6.056471e-01	5.212781e-01	4.822270e-01	4.036325e-01	3.300833e-01	250.120109	0.041527
3407e+01	...	-3.483038e+01	-1.093314e+01	-4.480774e+01	-2.836627e+00	-1.029540e+01	-2.604551e+00	-2.256568e+01	-1.543008e+01	0.000000	0.000000
30976e-01	...	-2.283949e-01	-5.423504e-01	-1.618463e-01	-3.545861e-01	-3.171451e-01	-3.269839e-01	-7.083953e-02	-5.295979e-02	5.600000	0.000000
42873e-02	...	-2.945017e-02	6.781943e-03	-1.119293e-02	4.097606e-02	1.659350e-02	-5.213911e-02	1.342146e-03	1.124383e-02	22.000000	0.000000
71390e-01	...	1.863772e-01	5.285536e-01	1.476421e-01	4.395266e-01	3.507156e-01	2.409522e-01	9.104512e-02	7.827995e-02	77.165000	0.000000
9499e+01	...	2.720284e+01	1.050309e+01	2.252841e+01	4.584549e+00	7.519589e+00	3.517346e+00	3.161220e+01	3.384781e+01	25691.160000	1.000000

Figure 24. Exploring the dataset after PCA transformation

7.2. PCA and its components

As aforementioned PCA was used in my dataset. This principal component analysis is a dimensionality reduction method, which is often used to reduce the dimensions of large datasets, by transforming tedious sets of variables into smaller ones that still preserve the initial data sets' main information.

By having a more manageable number of variables, it can be easier for the data to be handled and tackled with a view to processing it. Also, smaller sets are more manageable to explore and visualize by making so the analysis of the data points much easier and faster for machine learning algorithms. In general terms, the idea of PCA revolves around the reduction on the number of variables of a group of data, by simultaneously storing as much information as possible [2].

Principal components are new variables created by combining or blending the original variables in a linear fashion. These combinations are designed to ensure that the new variables are uncorrelated, while retaining most of the information from the original variables. For instance, in a scenario with 10-dimensional data, 10 principal components are generated. However, the goal of PCA is to maximize the information stored in the first component, followed by the maximum remaining information in subsequent components, until it approximates the illustration shown in the chart below.

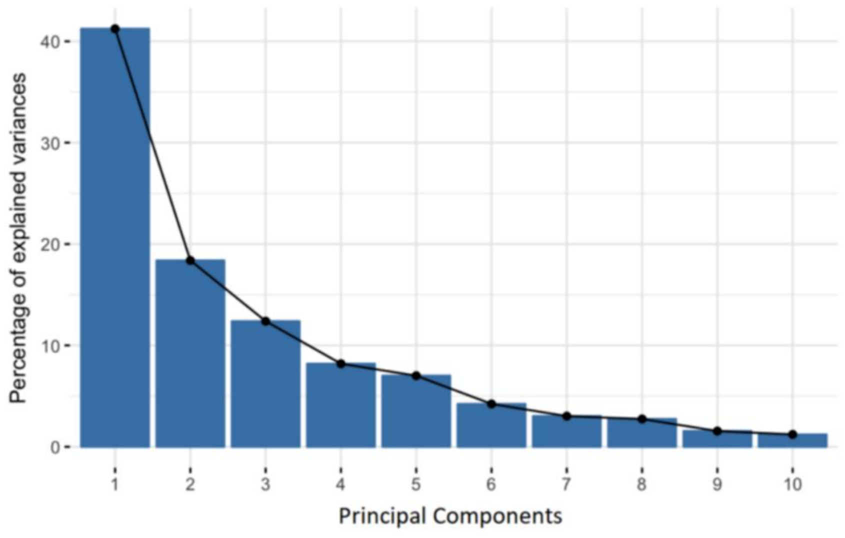


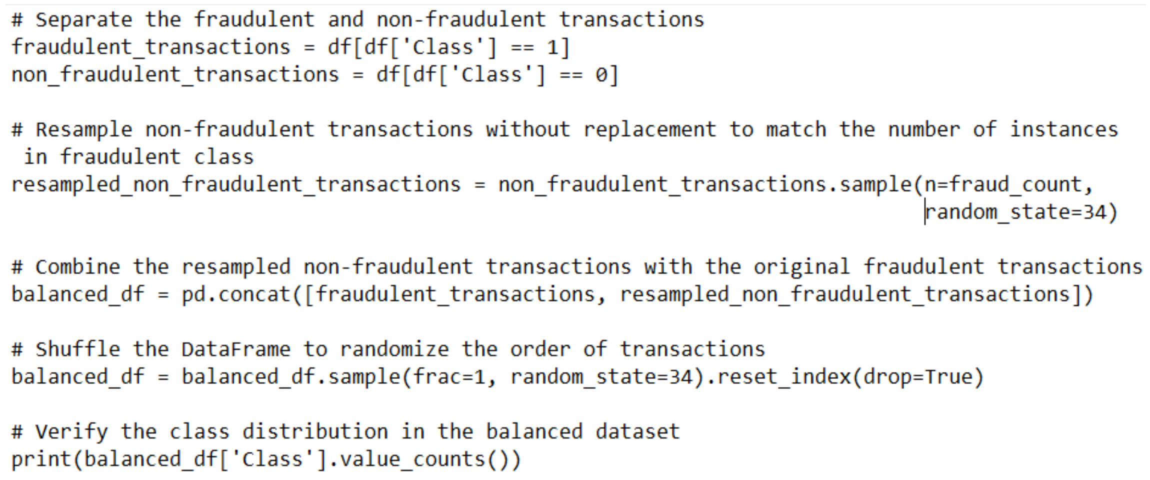
Figure 25. Principal components [builtin.com]

The organization of the information in the principal components in such a way allows to bring down the dimensionality without losing too much information, by removing low-information components and by considering the remaining ones as new variables. Since there are as many principal components as there are variables in the dataset, these components are designed to ensure that the first one captures the highest variance.

Geometrically, principal components represent the directions within the data that reveal the highest variance. The connection between variance and information is evident: a line with greater variance accommodates a wider spread of data points along it, signifying richer informational content. In essence, principal components act as new axes that offer the optimal perspective to analyze and interpret the data.

7.2.1. Handling the unbalanced data

Before working with the dataset, I handled the imbalanced data as follows:



```
# Separate the fraudulent and non-fraudulent transactions
fraudulent_transactions = df[df['Class'] == 1]
non_fraudulent_transactions = df[df['Class'] == 0]

# Resample non-fraudulent transactions without replacement to match the number of instances
 in fraudulent class
resampled_non_fraudulent_transactions = non_fraudulent_transactions.sample(n=fraud_count,
                                                                           random_state=34)

# Combine the resampled non-fraudulent transactions with the original fraudulent transactions
balanced_df = pd.concat([fraudulent_transactions, resampled_non_fraudulent_transactions])

# Shuffle the DataFrame to randomize the order of transactions
balanced_df = balanced_df.sample(frac=1, random_state=34).reset_index(drop=True)

# Verify the class distribution in the balanced dataset
print(balanced_df['Class'].value_counts())
```

Figure 26. Handling the unbalanced data

Nevertheless, Principal Component Analysis and Imbalanced Data Handling, are two distinct concepts in Machine Learning and data analysis.

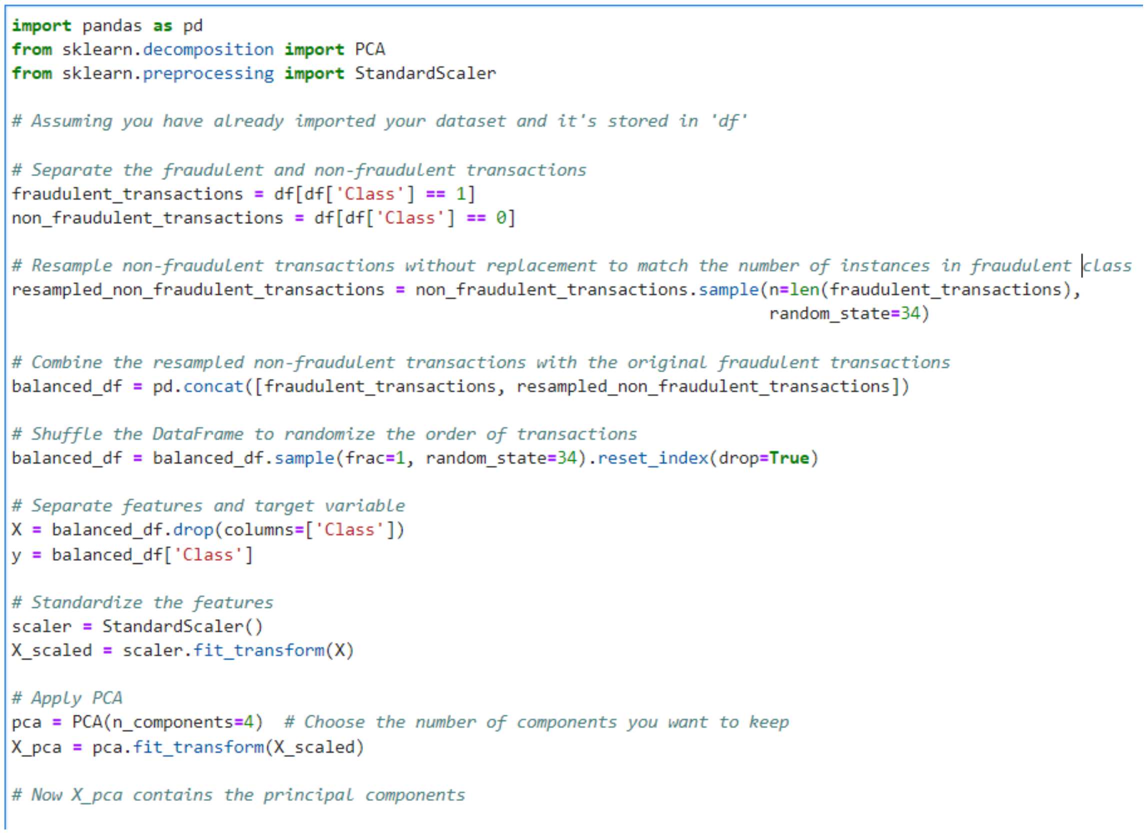
Imbalanced data arises when the classes within a dataset are not evenly distributed, meaning that one class may have notably more instances than another. Such an imbalance can result in biased models that exhibit poor performance, particularly in the minority class. Addressing this imbalance involves employing various techniques, including sampling methods such as minority class sampling and majority class subsampling, generating synthetic samples, utilizing alternative evaluation metrics like F1 score, precision, and recall, or using specialized algorithms designed to handle imbalanced data, such as Random Forest with class weights or gradient boosting.

So in principle, PCA and unbalanced data handling serve distinct purposes: PCA primarily focuses on dimensionality reduction, while methods for handling unbalanced data aim to address issues related to class imbalance in classification tasks.

7.2.2. Applying PCA to my dataset

To perform principal component analysis on the balanced dataset, after handling their imbalance, I followed these steps:

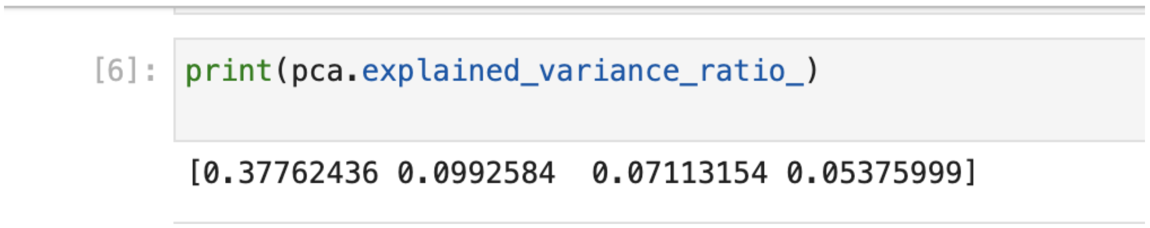
- Features and target separation, which helped in distinguishing the independent variables from the target one.
- Variable standardization: this helped to ensure that all features contributed evenly in the analysis. Standardization means scaling the features to have a mean of 0 and a standard deviation of 1.
- Application of PCA after the features were standardized, to reduce the dimensionality.



```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Assuming you have already imported your dataset and it's stored in 'df'

# Separate the fraudulent and non-fraudulent transactions
fraudulent_transactions = df[df['Class'] == 1]
non_fraudulent_transactions = df[df['Class'] == 0]

# Resample non-fraudulent transactions without replacement to match the number of instances in fraudulent class
resampled_non_fraudulent_transactions = non_fraudulent_transactions.sample(n=len(fraudulent_transactions),
                                                                           random_state=34)

# Combine the resampled non-fraudulent transactions with the original fraudulent transactions
balanced_df = pd.concat([fraudulent_transactions, resampled_non_fraudulent_transactions])

# Shuffle the DataFrame to randomize the order of transactions
balanced_df = balanced_df.sample(frac=1, random_state=34).reset_index(drop=True)

# Separate features and target variable
X = balanced_df.drop(columns=['Class'])
y = balanced_df['Class']

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply PCA
pca = PCA(n_components=4)  # Choose the number of components you want to keep
X_pca = pca.fit_transform(X_scaled)

# Now X_pca contains the principal components
```

Figure 27. Coding for applying PCA

After having applied PCA on my dataset, a validation process was performed so as to confirm it was correctly applied. Firstly, PCA calculated the explained variance ratio for each principal component.



```
[6]: print(pca.explained_variance_ratio_)

[0.37762436 0.0992584  0.07113154 0.05375999]
```

Figure 28. Validating PCA through the explained variance ratio

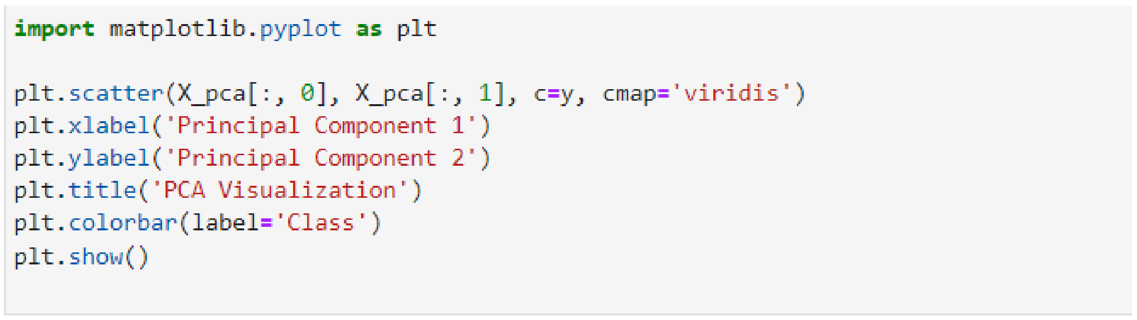
The outputted values 0.37762436, 0.0992584, 0.07113154 and 0.05375999 represent the explained variance ratio for each principal component.

First principal component (PC1): the value 0.37762436 demonstrated that the first principal component explained approximately 37.76% of the total variance in the original dataset, which meant that only 37.76% of the variability in the data could be captured by this single component.

Second principal component (PC2): the value 0.0992584 demonstrated that the second principal component explained approximately 9.93% of the total variance in the original dataset, meaning that only 9.93% of the variability in the data could be captured by this second component.

These two values were essential because they determined how much information would be stored when the dimensionality of the dataset was reduced by PCA. Based on these values of the explained variance ratio, I decided on the number of principal components that I was going to store.

Secondly, after calculating the explained variance ratio for each principal component, what followed was the visualization of the principal components. This type of validation showed how the data had been transformed into the new feature space.



```
import matplotlib.pyplot as plt

plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='viridis')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Visualization')
plt.colorbar(label='Class')
plt.show()
```

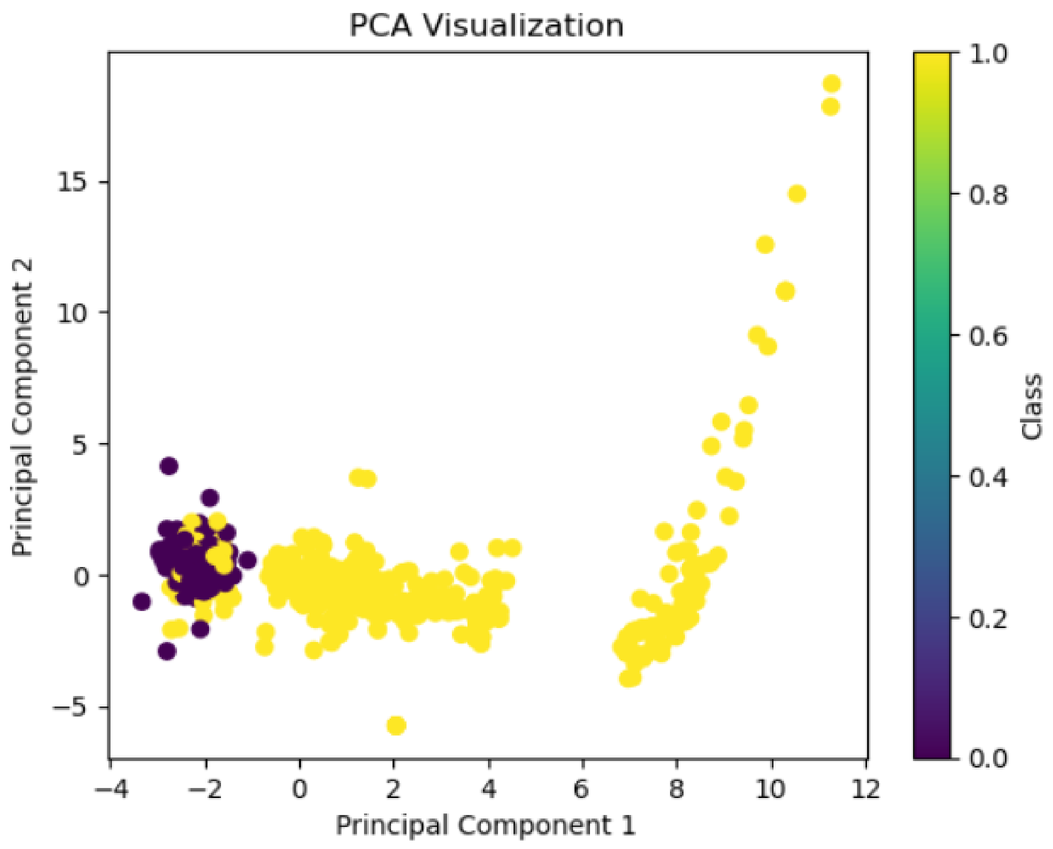
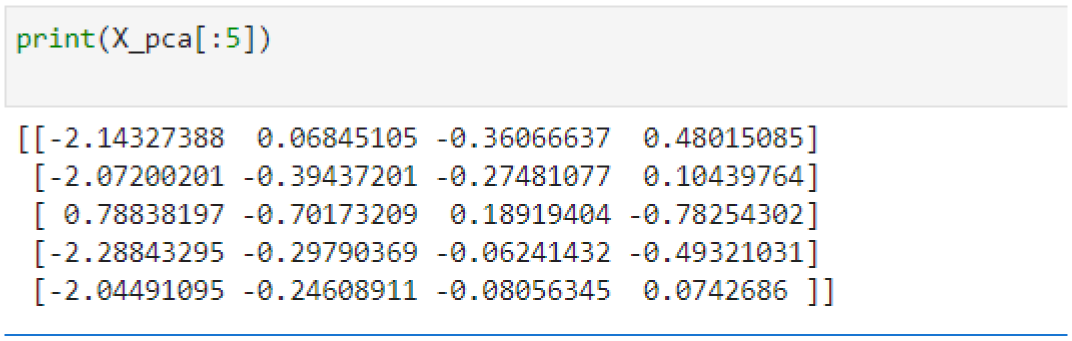


Figure 29. PCA visualization

After having done such modifications, first five lines of the transformed dataset were displayed.



```
print(X_pca[:5])
```

```
[[-2.14327388  0.06845105 -0.36066637  0.48015085]
 [-2.07200201 -0.39437201 -0.27481077  0.10439764]
 [ 0.78838197 -0.70173209  0.18919404 -0.78254302]
 [-2.28843295 -0.29790369 -0.06241432 -0.49321031]
 [-2.04491095 -0.24608911 -0.08056345  0.0742686 ]]
```

Figure 30. The dataset after the PCA application

Table 1. List of available features in the CCFD dataset

No.	Characteristic	Description
1	Bank account	Related to the account number
2	Ready to buy	Balance availability
3	Credit limit	Maximum credit capacity for the user
4	Card number	Card number
5	Transaction amount	The amount paid by the buyer
6	Transaction time	The time of the transaction
7	Transaction date	The date of the transaction
8	Currency code	The code of the used

		currency
9	Merchant category code	Merchant category code
10	Merchant number	Merchant number
11	Transaction country	Country where the transaction takes place
12	Transaction city	City where the transaction takes place
13	Approval code	The request authorization response, meaning approved or denied.

Table 2. Data characteristics

No.	Feature	Description
1	Time	Time expressed in seconds between the current transaction and the previous one
2	V1 – V28 attribute	These 28 columns show the PCA results to store the users’ data
3	Amount	Transaction amount

4	Binary class label	Binary classes, expressed by 0 for non-fraudulent classes and 1 for fraudulent ones

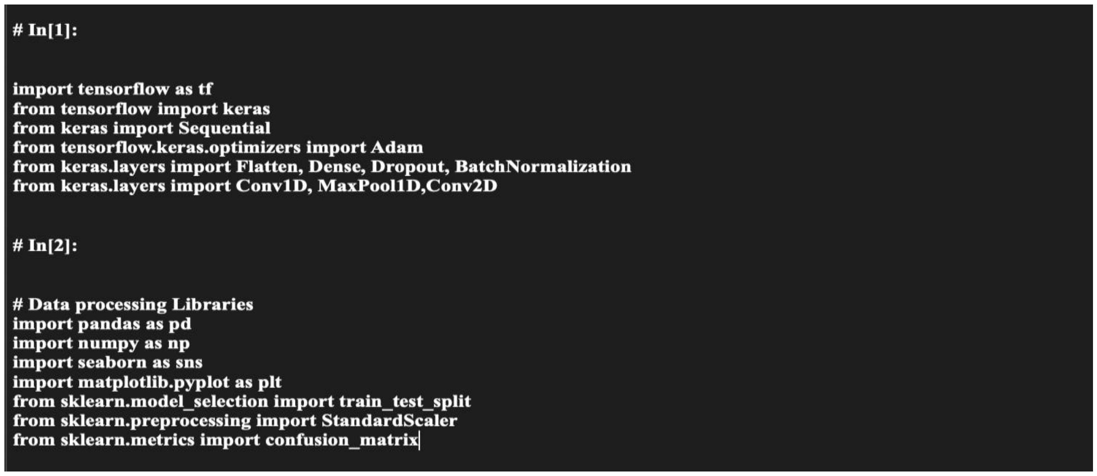
7.3 The steps that were followed to identify the data in the code

The approach that I chose to tackle my topic and to arrange the dataset is composed of a series of steps, which put together give a much better understanding on how the banking transactions work, what happens with them and how to eliminate the fraudulent data. In the following paragraphs I will briefly explain what I have used and the outputs that have been produced by the created model.

The code builds a Convolutional Neural Network model for fraud detection. CNNs are a type of ML model commonly used for image and sequence data. In this case they are applied on the ID data (sequential data). The architecture of the model is defined by using Keras. It includes convolutional layers with ReLU activation functions, batch normalization to ameliorate training stability and dropout layers to prevent overlapping. The final dense layer with a sigmoid activation function is responsible for making binary predictions (fraudulent or non-fraudulent). The model is then trained up until a certain time period.

Step 1: Library importing

The script initiates by importing essential Python libraries. TensorFlow and Keras are employed for constructing and training the neural network model. Additionally, various other libraries, including Pandas, NumPy, Seaborn, Matplotlib, and Scikit-Learn, are utilized for tasks such as data manipulation, visualization, and assessing the performance of the constructed model.

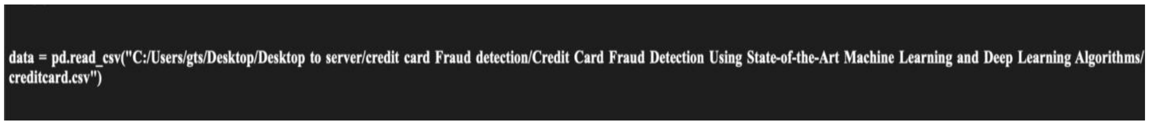


```
# In[1]:


import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from tensorflow.keras.optimizers import Adam
from keras.layers import Flatten, Dense, Dropout, BatchNormalization
from keras.layers import Conv1D, MaxPool1D,Conv2D


# In[2]:


# Data processing Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
```

Figure 31. Importing libraries in machine learning

Step 2: Data uploading

The pd.read_csv function is used to upload the CSV file that contains the credit card transaction data. This data is stored in a Pandas data frame called “data”. The data contains various attributes linked to the credit card transactions.



```
data = pd.read_csv("C:/Users/gts/Desktop/Desktop to server/credit card Fraud detection/Credit Card Fraud Detection Using State-of-the-Art Machine Learning and Deep Learning Algorithms/
creditcard.csv")
```

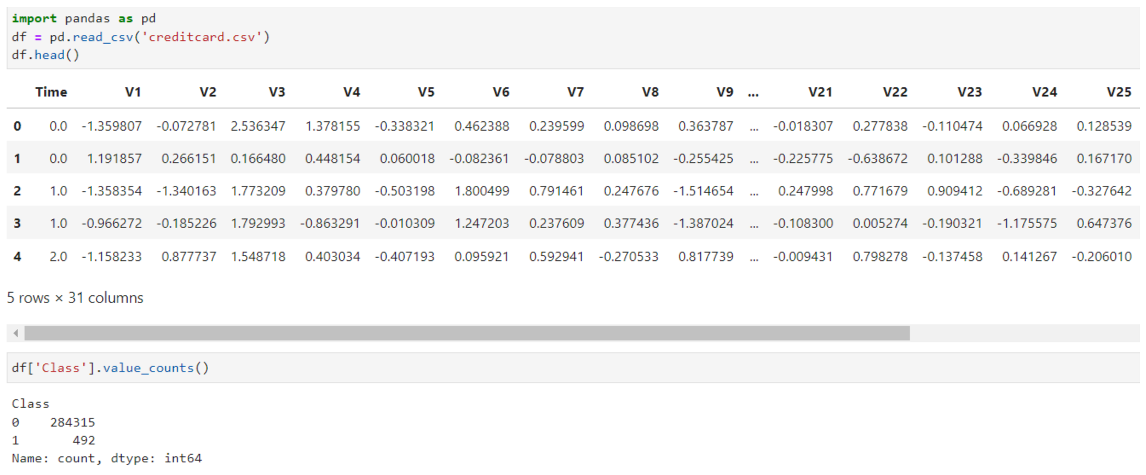
Figure 32. Uploading the ML data

Step 3. Exploratory data analysis (EDA)

The script performs basic exploratory analysis of the uploaded numerical dataset so as to discover patterns and main characteristics.

- Data.info() offers information about the DataFrame, including data types and non-null numerations
- Data.shape() gives the dimensions of the dataset, in terms of the number of rows and columns
- Data.describe() summarizes the basic statistics for each numeric column in the dataset

As aforementioned, after uploading the dataset and the corresponding libraries, I called "value_counts()" in order to get the total sum of the transactions that fall in each label category, where it resulted that there were 492 fraudulent ones and 284315 normal transactions.



```
import pandas as pd
df = pd.read_csv('creditcard.csv')
df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010

5 rows × 31 columns

```
df['Class'].value_counts()
```

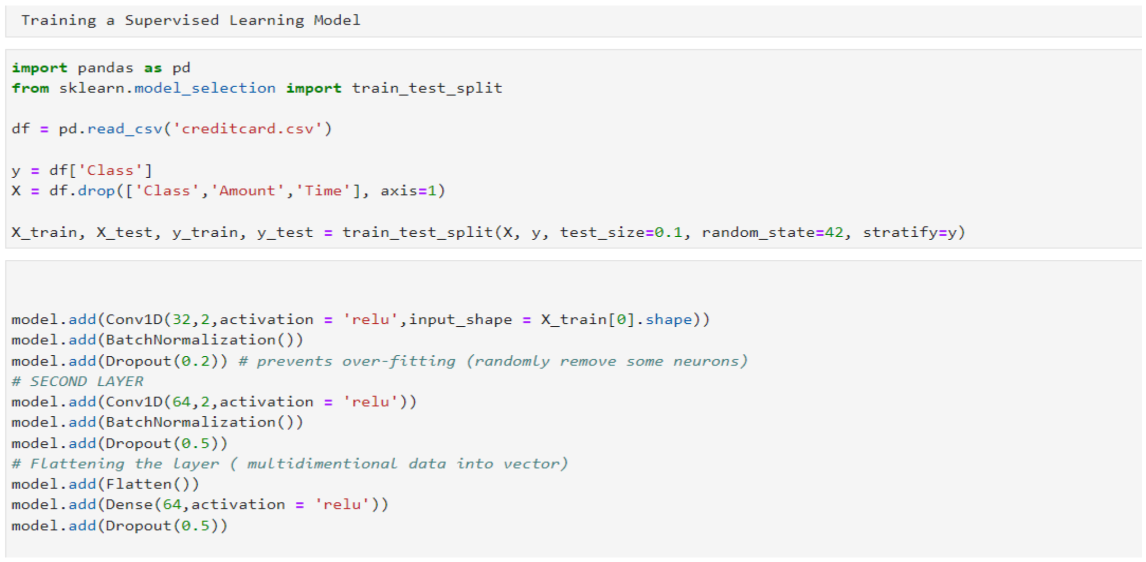
```
Class
0    284315
1       492
Name: count, dtype: int64
```

Figure 33. Exploratory data analysis EDA

In the cases when the class imbalance is a problem (i.e. different classes have different numbers of examples), it is important to balance to group of data. The minority class, being the one of fraud transactions, is randomly selected by selecting an equal number of non-fraudulent transactions. This balancing step helps to prevent the model from being biased towards the majority class.

Step 4: Data preprocessing

Preparing the data is a pivotal stage in machine learning, involving tasks such as distinguishing between features and labels, partitioning the data into training and testing sets, and implementing feature standardization. The standardization process is crucial for ensuring uniform scales across all features, facilitating the model's effective learning of patterns.



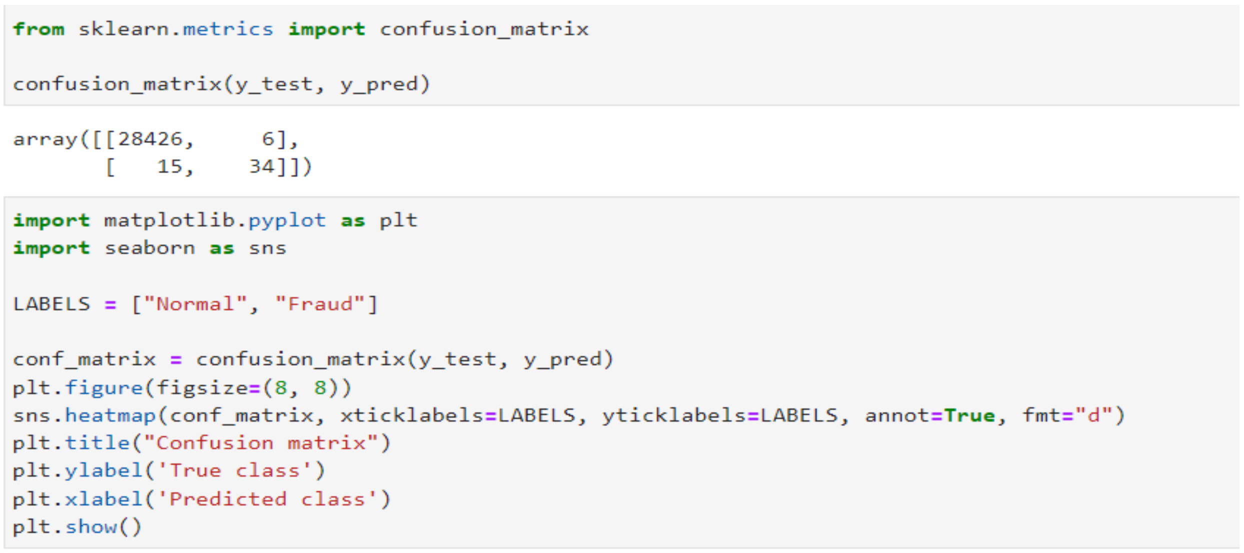
```
Training a Supervised Learning Model
```

```
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv('creditcard.csv')

y = df['Class']
X = df.drop(['Class','Amount','Time'], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42, stratify=y)
```

```
model.add(Conv1D(32,2,activation = 'relu',input_shape = X_train[0].shape))
model.add(BatchNormalization())
model.add(Dropout(0.2)) # prevents over-fitting (randomly remove some neurons)
# SECOND LAYER
model.add(Conv1D(64,2,activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
# Flattening the layer ( multidimentional data into vector)
model.add(Flatten())
model.add(Dense(64,activation = 'relu'))
model.add(Dropout(0.5))
```

Figure 34. Training the data through a supervised learning method.

The trained model generates predictions to assess its accuracy, quantifying the number of true positives, true negatives, false positives, and false negatives through the use of a confusion matrix.



```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test, y_pred)
```

```
array([[28426,     6],
       [   15,    34]])
```

```
import matplotlib.pyplot as plt
import seaborn as sns

LABELS = ["Normal", "Fraud"]

conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 8))
sns.heatmap(conf_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt="d")
plt.title("Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
```

Figure 35. Confusion matrix

This is the output produced:

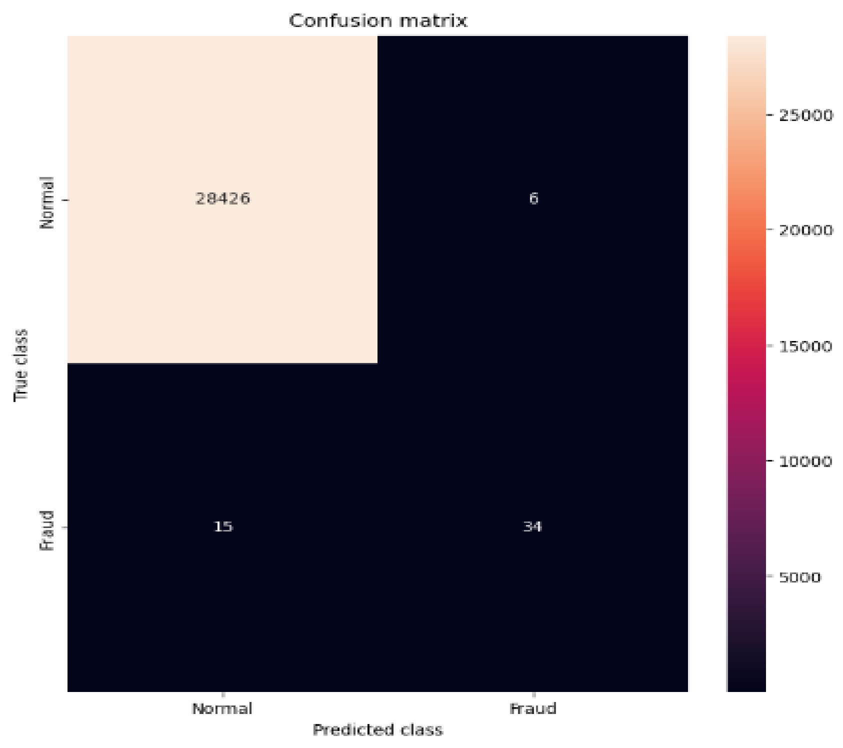


Figure 36. Output of confusion matrix

The model continues to get trained up until a certain time period in order to improve the logistic regression.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(class_weight='balanced')
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
y_pred
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
import matplotlib.pyplot as plt
import seaborn as sns
LABELS = ["Normal", "Fraud"]
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(12, 12))
sns.heatmap(conf_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt="d")
plt.title("Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
```

Figure 37. Model training

It outputs a changed confusion matrix.

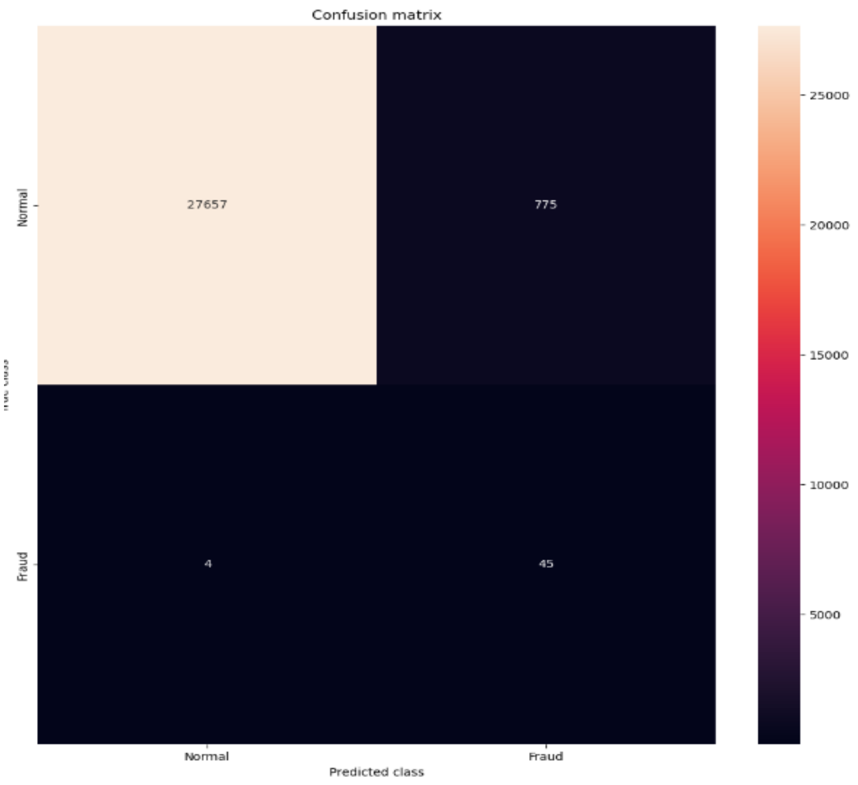
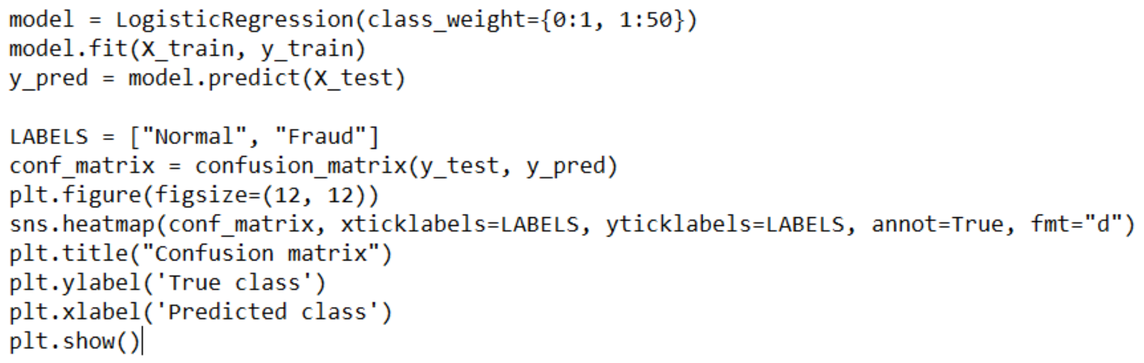


Figure 38. Changed confusion matrix

The process continues and the weights are updated again and this produces a new output.



```
model = LogisticRegression(class_weight={0:1, 1:50})
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

LABELS = ["Normal", "Fraud"]
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(12, 12))
sns.heatmap(conf_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt="d")
plt.title("Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
```

Figure 39. Updating weights in confusion matrix

Model evaluation:

- To assess the model's performance, the script visualizes learning curves, providing insights into how accuracy and training loss evolve over time.
- The accuracy of the model on the training data is scrutinized to gauge its effectiveness in learning from the training set.
- A confusion matrix is computed on the test data, offering a breakdown of true positives, true negatives, false positives, and false negatives predicted by the model.
- The script calculates "precision" and "recall," pivotal metrics for appraising binary classification model performance. Leveraging the Scikit-Learn metrics library, "precision" quantifies the accuracy of positive predictions, while "recall" gauges the model's ability to capture actual positive cases.
- Individual prediction: the script shows how to make predictions for an individual transaction. It scales the features of the transaction and feeds them to the trained model to classify it as a fraudulent transaction or not.

CONCLUSIONS

Studying fraudulent data using Python and Machine Learning can bring about many important conclusions for the identification and prevention of such illicit activities. This type of analysis consists of several different steps, being:

- Data preparation
- Data exploration: refers to the analysis of the frequencies and the variables to understand the characteristics of the data. This allows the identification of any patterns emerging from the data.
- Choosing the model: choosing the proper model is very important. Models like: Random Forest, Support Vector Machines (SVM) and Neural networks are often used to identify dangerous schemes.
- Handling fraudulent data: in the case of fraudulent data in the dataset, different approaches can be followed to handle them. Deleting rows, columns or other methods can be helpful to solve this issue.
- Modeling and uncertainty handling: uncertainty can be handled with various modeling methods, including cross-validation and hyperparameter research to ensure a good model.

Conclusions about this model:

- Essential factors influencing the performance of the fraud detection model include credit card fraud detection itself, the number of features, the number of transactions, and the correlation between features.
- Deep learning (DL) methods, specifically Convolutional Neural Networks (CNN) and their associated layers, are traditionally linked with text processing and form the core of the model. Leveraging these methods for credit card fraud

detection has demonstrated superior performance compared to conventional algorithms.

- Using Convolutional Neural Network in this project for the detection of deceitful transactions demonstrated that we have explored a very accurate machine learning method.

Preventing fraudulent transactions:

There are some actions that can be taken in order to reduce the high levels of fraudulent transactions, like:

- Identity verification: requiring a strong identity verification process for the users, including two-factor authentication.
- Urge users to regularly change and refresh their passwords.
- Activity monitoring: by setting up a monitoring system for users' activity it can detect suspicious and unusual activity.
- Use advanced technology and analytics to identify suspicious transaction patterns.
- Risk analysis: use risk analysis technology to evaluate the potential level of threat for each transaction. Also, by being aware of possible credit card usage patterns that are common for fraudulent transactions.

REFERENCES

[1] A. Kaleel, "Research Gate," 6 December 2023. [Online]. Available: https://www.researchgate.net/publication/377417277_Credit_Card_Fraud_Detection_and_Identification_using_Machine_Learning_Techniques.

[2] H. A. Lynne J Williams, "Research gate," 24 July 2010. [Online]. Available: https://www.researchgate.net/publication/227644862_Principal_Component_Analysis. [Accessed 25 January 2024].

[3] S. G. Vaishnavi Nath Dornadula, "Research gate," 12 January 2019. [Online]. Available: https://www.researchgate.net/publication/339543849_Credit_Card_Fraud_Detection_using_Machine_Learning_Algorithms. [Accessed 25 January 2024].

[4] Y. Z. C. H. Yong Fang, "Research gate," 16 January 2019. [Online]. Available: https://www.researchgate.net/publication/336062599_Credit_Card_Fraud_Detection_Based_on_Machine_Learning. [Accessed 26 January 2024].

[5] A. B. Said Jadid Abdulkadir, "Research gate," 14 July 2019. [Online]. Available: https://www.researchgate.net/profile/Abdullateef-Balogun/publication/334319562_Performance_Analysis_of_Feature_Selection_Methods_in_Software_Defect_Prediction_A_Search_Method_Approach/links/5d24782ba6fdcc2462ce38ca/Performance-Analysis-of-Feature-Selectio. [Accessed 26 January 2024].

[6] P. A. P. B. A. S. G. Bernardo Branco, "Research gate," 14 February 2020. [Online]. Available: https://paperswithcode.com/paper/interleaved-sequence-rnns-for-fraud-detection. [Accessed 20 January 2024].

[7] B. O. I. B. Samira Douzi, "Research gate," 04 January 2021. [Online]. Available: https://www.researchgate.net/profile/Samira-Douzi/publication/350691336_Credit_Card_Fraud_Detection_Model_Based_on_LSTM_Recurrent_Neural_Networks/links/60f18b0ffb568a7098b2a188/Credit-Card-Fraud-Detection-Model-Based-on-LSTM-Recurrent-Neural-Networks.pdf. [Accessed 01 February 2024].

[O. A. O. E. F. Cartella, "Semantic Scholar," 20 January 2021. [Online]. Available:

[8] https://www.semanticscholar.org/paper/Adversarial-Attacks-for-Tabular-Data:-Application-Cartella-Anuncia%C3%A7%C3%A3o/ea33122e1599d9c35323ec2af6a413d125c09a31/figure/1. [Accessed 15 January 2024].

[9] X. Z. S. R. J. S. Kaiming He, "Research gate," 14 January 2019. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf. [Accessed 20 January 2024].

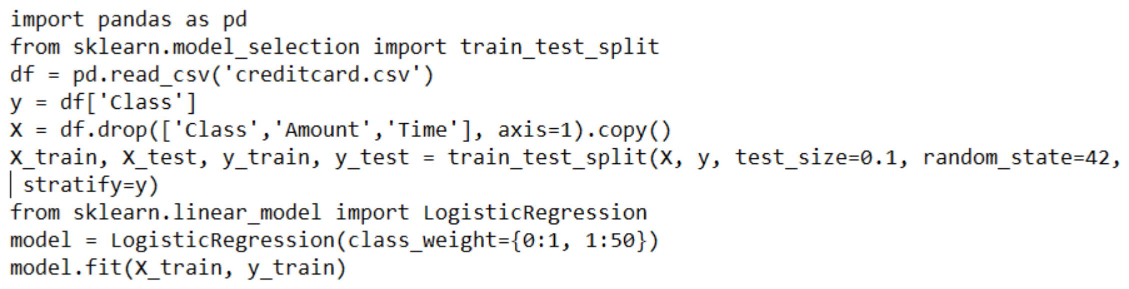
[10] S. S. L. Amol C Adamuthe, "Research gate," 20 December 2020 . [Online]. Available: https://www.researchgate.net/publication/347901019_Malware_Classification_with_Improved_Convolutional_Neural_Network_Model. [Accessed 20 January 2024].

[11] J. F. Saeedeh Momtazi, "Research gate," 20 November 2020. [Online]. Available: https://www.researchgate.net/publication/345896338_Ensemble_of_deep_sequential_models_for_credit_card_fraud_detection. [Accessed 02 January 2024].

[12] H. C. R. Z. Xinyi Hu, "Research gate," 21 September 2019. [Online]. Available: https://www.researchgate.net/publication/339821451_Short_Paper_Credit_Card_Fraud_Detection_using_LightGBM_with_Asymmetric_Error_Control. [Accessed 26 November 2023].

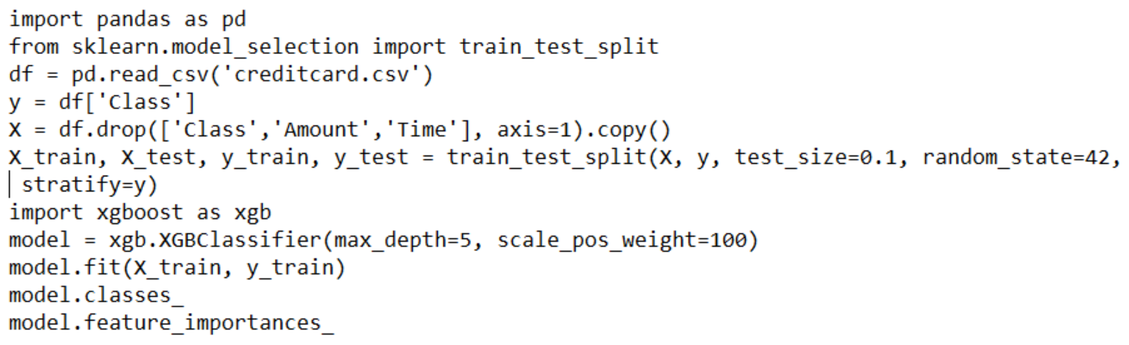
APPENDIX A

SOME CODING ELEMENTS



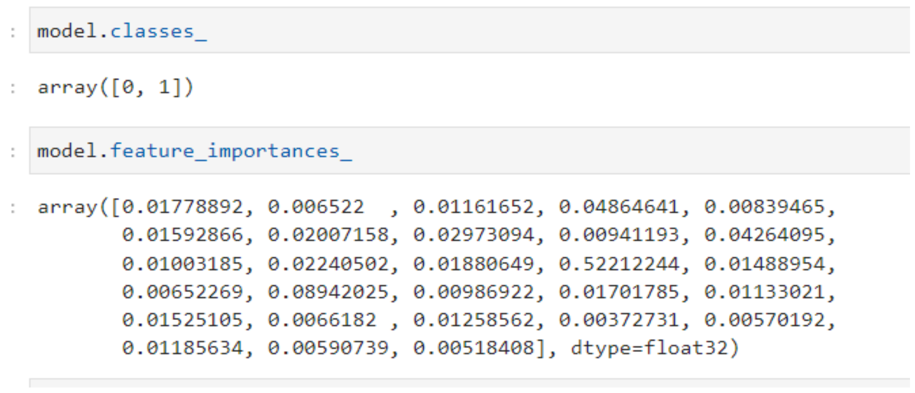
```
import pandas as pd
from sklearn.model_selection import train_test_split
df = pd.read_csv('creditcard.csv')
y = df['Class']
X = df.drop(['Class','Amount','Time'], axis=1).copy()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42,
 stratify=y)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(class_weight={0:1, 1:50})
model.fit(X_train, y_train)
```

Figure 40. Interpreting the logistic regression model



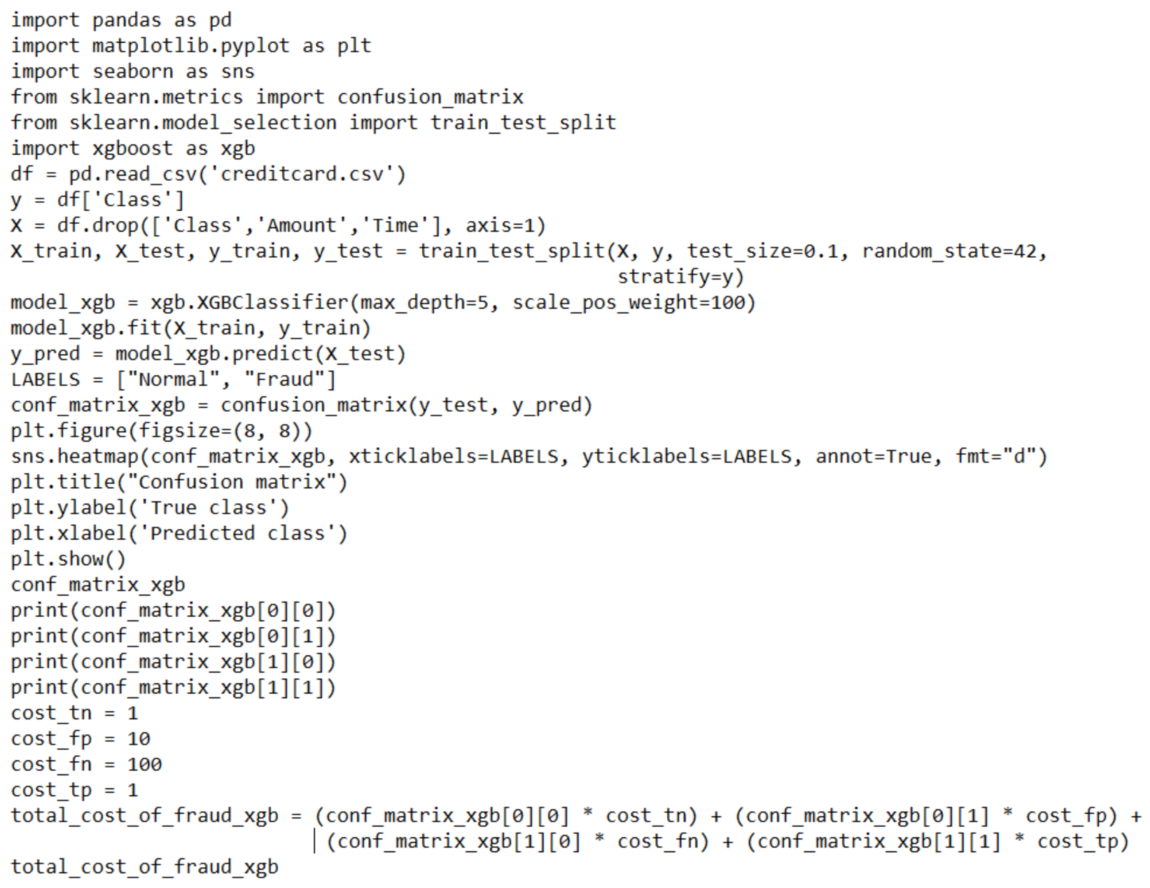
```
import pandas as pd
from sklearn.model_selection import train_test_split
df = pd.read_csv('creditcard.csv')
y = df['Class']
X = df.drop(['Class','Amount','Time'], axis=1).copy()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42,
 stratify=y)
import xgboost as xgb
model = xgb.XGBClassifier(max_depth=5, scale_pos_weight=100)
model.fit(X_train, y_train)
model.classes_
model.feature_importances_
```

Figure 41. Interpreting the XGBoost Model



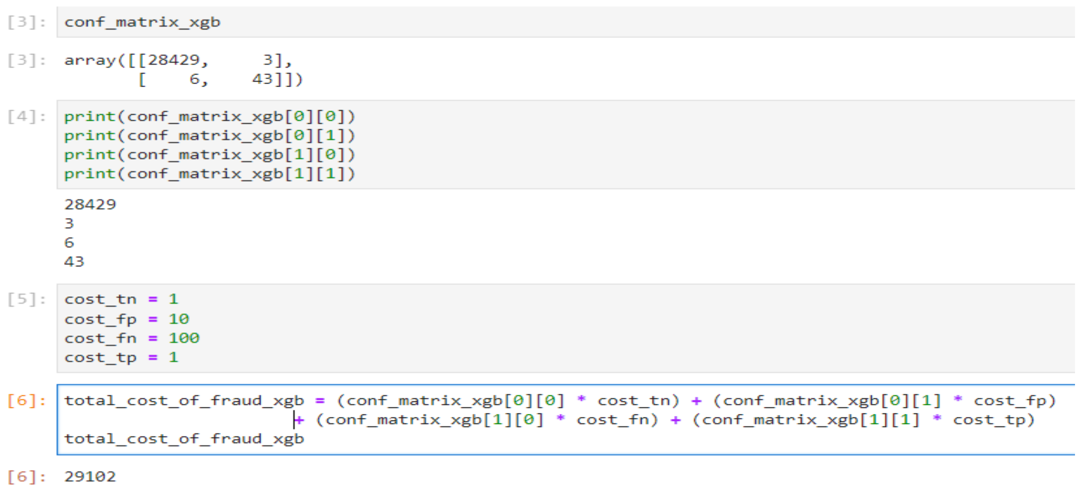
```
model.classes_

array([0, 1])

model.feature_importances_

array([0.01778892, 0.006522  , 0.01161652, 0.04864641, 0.00839465,
       0.01592866, 0.02007158, 0.02973094, 0.00941193, 0.04264095,
       0.01003185, 0.02240502, 0.01880649, 0.52212244, 0.01488954,
       0.00652269, 0.08942025, 0.00986922, 0.01701785, 0.01133021,
       0.01525105, 0.0066182 , 0.01258562, 0.00372731, 0.00570192,
       0.01185634, 0.00590739, 0.00518408], dtype=float32)
```

Figure 42. Output of using XGB classifier



```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
import xgboost as xgb
df = pd.read_csv('creditcard.csv')
y = df['Class']
X = df.drop(['Class','Amount','Time'], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42,
                                                    stratify=y)
model_xgb = xgb.XGBClassifier(max_depth=5, scale_pos_weight=100)
model_xgb.fit(X_train, y_train)
y_pred = model_xgb.predict(X_test)
LABELS = ["Normal", "Fraud"]
conf_matrix_xgb = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 8))
sns.heatmap(conf_matrix_xgb, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt="d")
plt.title("Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
conf_matrix_xgb
print(conf_matrix_xgb[0][0])
print(conf_matrix_xgb[0][1])
print(conf_matrix_xgb[1][0])
print(conf_matrix_xgb[1][1])
cost_tn = 1
cost_fp = 10
cost_fn = 100
cost_tp = 1
total_cost_of_fraud_xgb = (conf_matrix_xgb[0][0] * cost_tn) + (conf_matrix_xgb[0][1] * cost_fp) +
                          (conf_matrix_xgb[1][0] * cost_fn) + (conf_matrix_xgb[1][1] * cost_tp)
total_cost_of_fraud_xgb
```

Figure 43. Performance metrics for fraud detection (the cost of misclassification)



```
[3]: conf_matrix_xgb

[3]: array([[28429,     3],
            [    6,    43]])

[4]: print(conf_matrix_xgb[0][0])
     print(conf_matrix_xgb[0][1])
     print(conf_matrix_xgb[1][0])
     print(conf_matrix_xgb[1][1])

     28429
     3
     6
     43

[5]: cost_tn = 1
     cost_fp = 10
     cost_fn = 100
     cost_tp = 1

[6]: total_cost_of_fraud_xgb = (conf_matrix_xgb[0][0] * cost_tn) + (conf_matrix_xgb[0][1] * cost_fp)
                             + (conf_matrix_xgb[1][0] * cost_fn) + (conf_matrix_xgb[1][1] * cost_tp)
     total_cost_of_fraud_xgb

[6]: 29102
```

Figure 44. Cost of misclassification

And this is the confusion matrix:

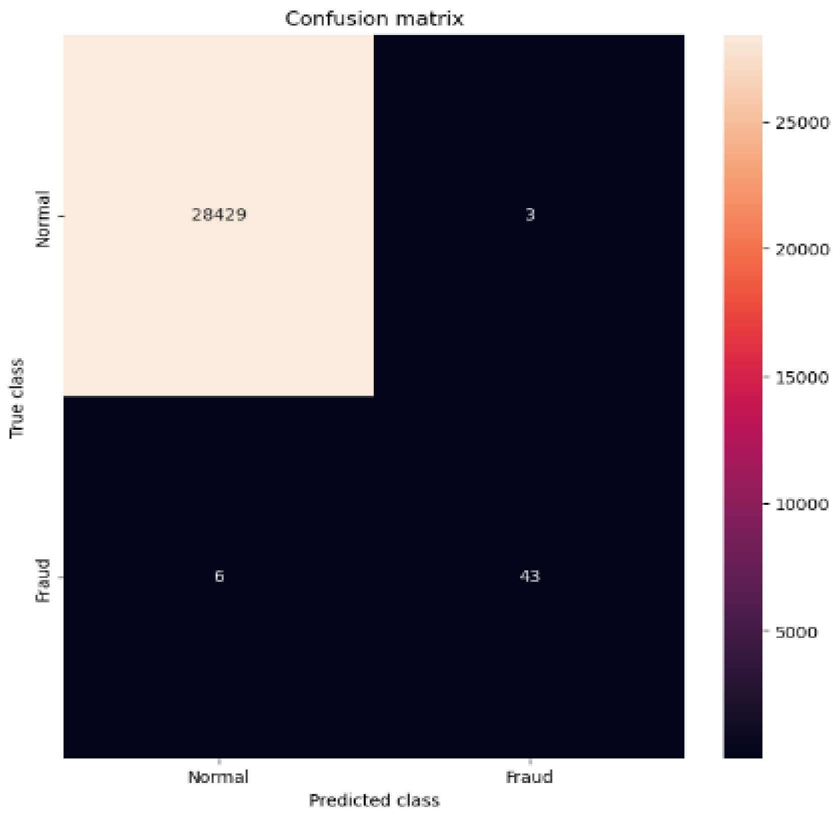
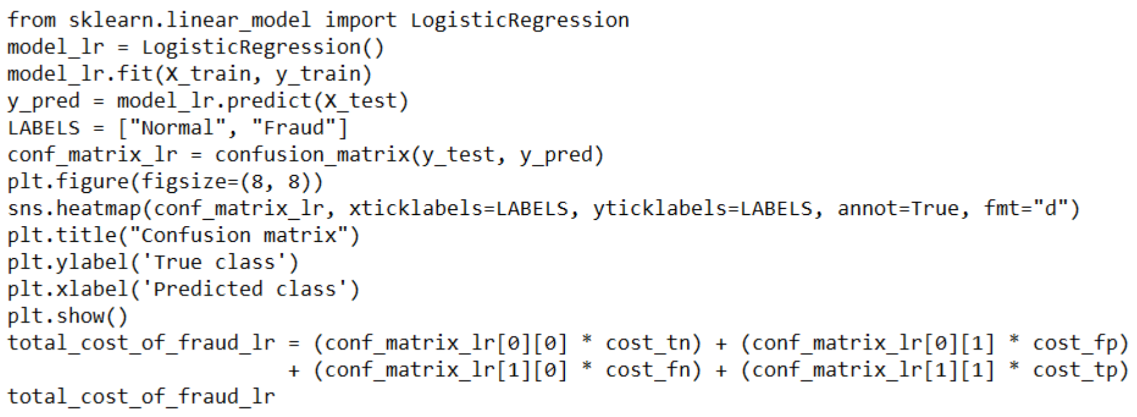
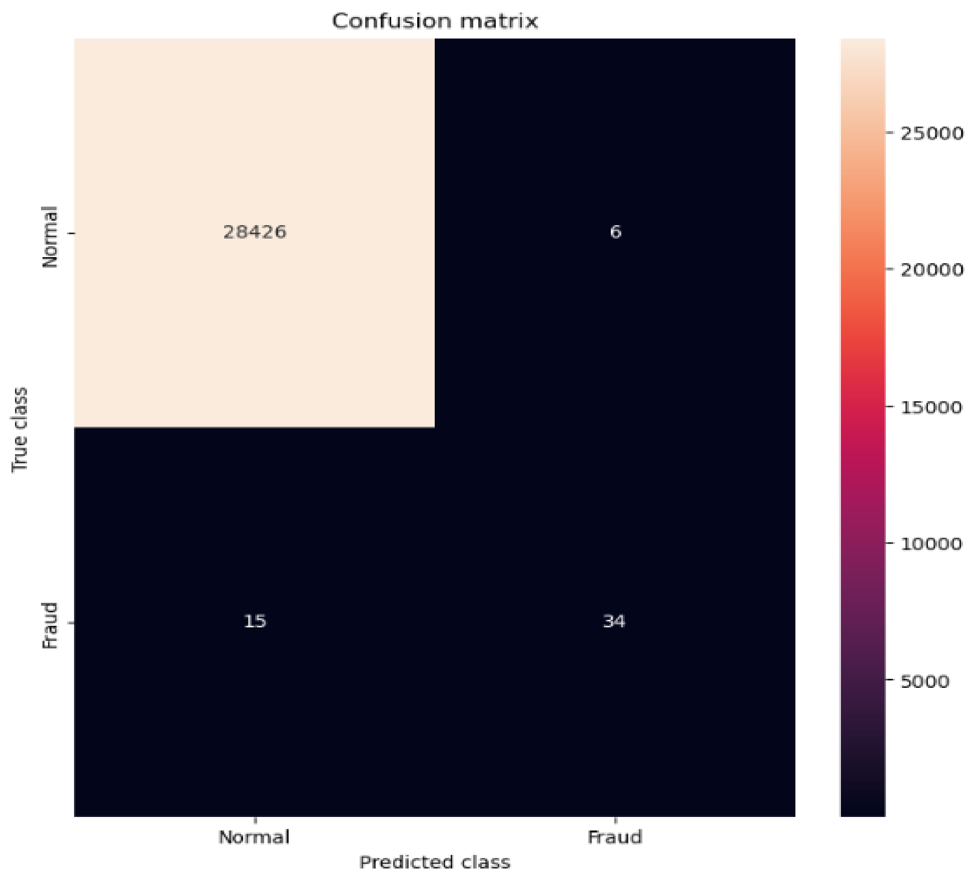


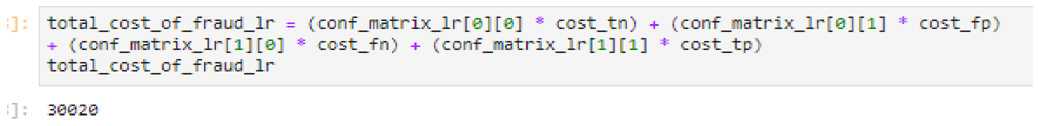
Figure 45. Confusion matrix XGB classifier



```
from sklearn.linear_model import LogisticRegression
model_lr = LogisticRegression()
model_lr.fit(X_train, y_train)
y_pred = model_lr.predict(X_test)
LABELS = ["Normal", "Fraud"]
conf_matrix_lr = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 8))
sns.heatmap(conf_matrix_lr, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt="d")
plt.title("Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
total_cost_of_fraud_lr = (conf_matrix_lr[0][0] * cost_tn) + (conf_matrix_lr[0][1] * cost_fp)
                       + (conf_matrix_lr[1][0] * cost_fn) + (conf_matrix_lr[1][1] * cost_tp)
total_cost_of_fraud_lr
```

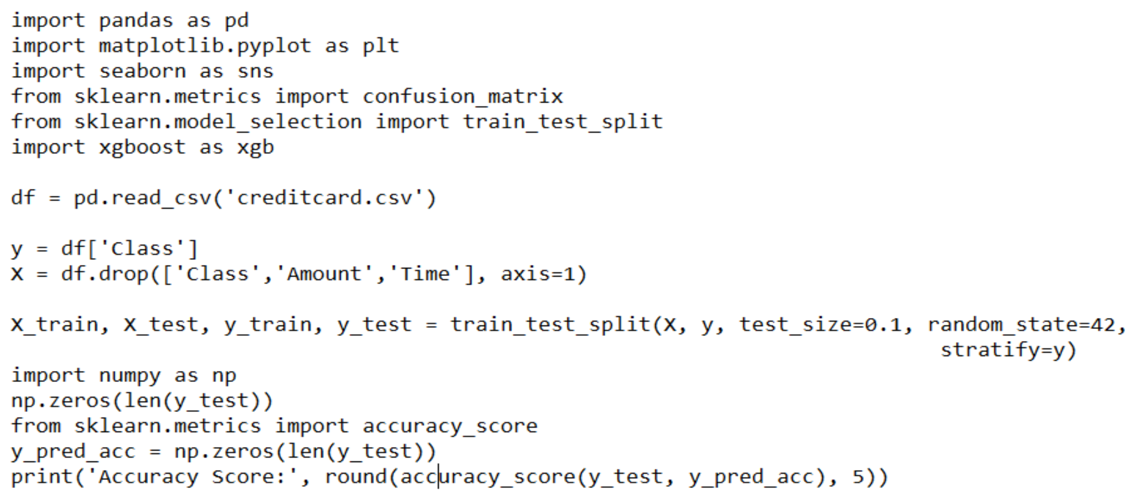
Figure 46. Code for changed cost of fraudulence





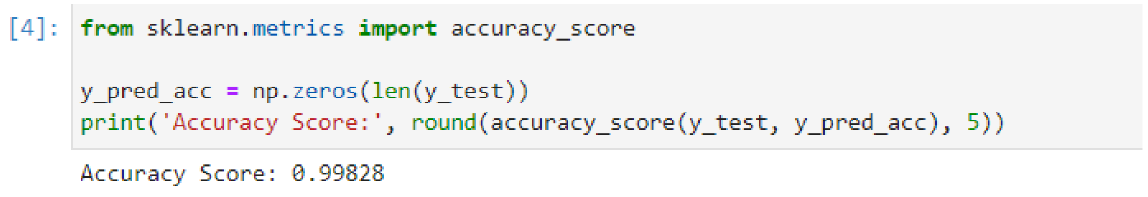
```
]:  total_cost_of_fraud_lr = (conf_matrix_lr[0][0] * cost_tn) + (conf_matrix_lr[0][1] * cost_fp)
    + (conf_matrix_lr[1][0] * cost_fn) + (conf_matrix_lr[1][1] * cost_tp)
    total_cost_of_fraud_lr

]:  30020
```

Figure 47. Matrix for changed cost



```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
import xgboost as xgb

df = pd.read_csv('creditcard.csv')

y = df['Class']
X = df.drop(['Class','Amount','Time'], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42,
                                                                          stratify=y)

import numpy as np
np.zeros(len(y_test))
from sklearn.metrics import accuracy_score
y_pred_acc = np.zeros(len(y_test))
print('Accuracy Score:', round(accuracy_score(y_test, y_pred_acc), 5))
```

Figure 48. Calculating the accuracy score



```
[4]: from sklearn.metrics import accuracy_score

     y_pred_acc = np.zeros(len(y_test))
     print('Accuracy Score:', round(accuracy_score(y_test, y_pred_acc), 5))

     Accuracy Score: 0.99828
```

Figure 49. Output of accuracy calculation