Czech University of Life Sciences Prague

Faculty of Environmental Sciences

Department of Information Technologies

Bachelor Thesis

ETL processing using Machine Learning

Shlapak Vladislav

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Environmental Sciences

BACHELOR THESIS ASSIGNMENT

Vladislav Shlapak

Environmental Data Science
Informatics

Thesis title

ETL processing using machine learning

Objectives of thesis

The main objective of the thesis is to propose, develop and evaluate a machine learning model that facilitates ETL process for a chosen data set.
Partial objectives include:
- Analysis of available information sources regarding machine learning and ETL processing
- Choose a testing data set
- Propose and develop machine learning model for ETL processing
- Analyze and evaluate the suitability of the model based on selected metrics

Methodology

The thesis will consist of two parts. The methodology of the theoretical part is based on analyzing available literary and other information sources. In the practical part, a machine learning model will be proposed and developed using Python programming language, including additional Python packages specialized for data processing and machine learning (Numpy, Pandas, Scikit-learn etc.). Model will be trained and validated using chosen dataset. Metrics regarding the model's performance and suitability will be measured and analyzed.

The proposed extent of the thesis

40-50

Keywords

machine learning, ETL processing, data analysis, data processing, data mining, Python, model perfomance analysis

Recommended information sources

DASGUPTA, N. *Practical big data analytics : hands-on techniques to implement enterprise analytics and machine learning using hadoop, spark, NoSQL and R.* Birmingham: Packt, 2018. ISBN 978-1783554393.

GÉRON, A. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems.* Beijing ; Boston ; Farnham ; Sevastopol ; Tokyo: O'Reilly, 2019. ISBN 978-1-492-03264-9.

HILL, T. – LEWICKI, P. *Statistics : methods and applications : a comprehensive reference for science, industry, and data mining.* Tulsa: StatSoft, 2006. ISBN 1-884233-59-7.

TOOMEY, D. *Jupyter for data science : exploratory analysis, statistical modeling, machine learning, and data visualization with Jupyter.* Birmingham: Packt, 2017. ISBN 978-1-78588-007-0.

Expected date of thesis defence

2022/23 SS – FES

The Bachelor Thesis Supervisor

Ing. Jan Pavlík

Supervising department

Department of Information Technologies

Electronic approval: 14. 7. 2022	Electronic approval: 29. 7. 2022
doc. Ing. Jiří Vaněk, Ph.D.	**prof. RNDr. Vladimír Bejček, CSc.**
Head of department	Dean

Prague on 12. 10. 2022

Declaration

I declare that I have worked on my bachelor thesis titled "ETL processing using Machine Learning" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break any copyrights.

In Prague on date of submission ___________________________

Acknowledgement

I would like to thank Pavlík Jan, Markonis Ioannis

ETL processing using Machine Learning

Abstract

The twenty-first century is the age of the information boom. Technology advances at a rapid pace, but so does the amount of information. A person today receives more information per day than our ancestors did in decades. The information must be stored and processed to obtain knowledge. Decision-making in many industries depends on the timely acquisition of knowledge from information. The faster and better the processing and analysis of information, for example, the more money it will bring to the business. Today, one of the most popular ways to store large amounts of data is the data warehouse concept. And the process of extract, transformation and loading is called the ETL process. This stage is the main one for getting started with information, but it is also the most time-consuming and error-prone. To increase the speed and accuracy of information processing, this work uses a deep machine learning model that will classify the type of trader's transaction by fields. I will explain why a deep learning model is needed to solve this problem and why automation of the ETL process is the next step in developing a more reliable and up-to-minute data management system.

Keywords: data science, mlps, classification, supervise, etl, deep learning, keras api, tensorflow, transactions, trading, python

ETL zpracování pomocí strojového učení

Abstrakt

Dvacáté první století je dobou informačního boomu. Technologie postupuje rychlým tempem, a roste množství informací. Každý člověk dnes přijímá více informací za den, než naši předkové za desítky let. Informace musí být uloženy a zpracovány pro získání znalostí. Rozhodování v mnoha odvětvích závisí na včasném získávání znalostí z informací. Rychlost a kvalita zpracování a analýzy dat například může podniku zvýšit zisky. Dnes je jedním z nejoblíbenějších způsobů ukládání velkého množství dat koncept datového skladu. Proces extrahování, transformace a načítání nese anglickou zkratku ETL. Tato fáze je hlavní pro začátek práce s informacemi, ale je také časově nejnáročnější a nejnáchylnější k chybám. Pro zvýšení rychlosti a přesnosti zpracování informací využívá tato bakalářská práce model hlubokého strojového učení, který bude klasifikovat typ transakce obchodníka podle polí. V práci je vysvětleno, proč je k vyřešení tohoto problému potřeba model hlubokého učení a jakým způsobem automatizace procesu ETL vede k dalšímu vývoji spolehlivějšího a aktuálnějšího systému správy dat.

Klíčová slova: zpracování dat, MLP, klasifikace, ETL, hluboké učení, Keras API, TensorFlow, transakce, obchodování, Python

Table of content

1 Introduction

Data is information output by a sensing device or organ that includes both useful and irrelevant or redundant information and must be processed to be meaningful. [1] In other words, the data is a unit of information. Nowadays, data are moving everywhere: when you open your eyes, you start getting information all around. In this thesis, I will focus on data as an essential part of all business, and technological processes. This data includes text information such as bills, detailed transactions, etc.

Google currently processes over 20 petabytes of data per day through an average of 100,000 MapReduce jobs spread across its massive computing clusters. [2] As we know technology is growing every moment and 20 petabytes in 2008 now could be more than 320 petabytes based on Moore's law. That huge amount of data, and this data comes through many types of sources. Collecting data is one initial part of using data. The collecting data could be divided into two groups based on type:

1. Quantitative data – numeric data.
2. Qualitative data – non-numeric data.

We have a different approach to collecting it at the start for each of the two groups. The data should be getting, processed, and used. This process is named the ETL process (extract, transform, load).

The ETL process is the most time-consuming and costly. For example, a data warehouse has two main types of sources:

1. Operation system: Enterprise Resource Planning (ERP) system, Customer Relationship Management systems (CRM), and On-Line Transaction Processing (OLTP) system.
2. External system, based on the company decision.

One of the primary challenges of managing large datasets is the need to add each record to the database. With billions of data rows, this process can be incredibly time-consuming. However, it is crucial to ensure that every data point is accurately stored to enable meaningful analysis and insights.

In addition to adding data to the database, it is essential to classify it based on its type. This classification can be useful in identifying patterns and trends in the data, which can inform decision-making processes. However, this classification process can also be time-consuming and resource-intensive, particularly when dealing with vast amounts of data.

Another critical step in data processing is removing duplicate entries. Duplicate data can skew analysis results and create misleading insights, making it crucial to identify and remove such records. Additionally, it is essential to merge data points that refer to the same variable, ensuring that the resulting dataset is accurate and complete.

Despite the importance of these steps, they are not without their challenges. Time constraints and resource limitations can make it challenging to complete these tasks thoroughly, potentially leading to inaccuracies in the final output data. Therefore, it is critical to leverage machine learning algorithms and data warehouses to streamline these processes, ensuring that accurate and meaningful insights can be obtained from vast amounts of data.

The transformation part of the ETL process rises different problems:

1. Schema-level problems – naming conflict.
2. Record-level problems – by the year, by the day.
3. Value-level problems - date formats, key management words. [3]

In this thesis to decrease cost and time-consuming as well as the number of mistakes I will use a machine learning model for the transformation part.

2 Objectives and Methodology

2.1 Objectives

The main objective of the thesis is to propose, develop and evaluate a machine learning model that facilitates the ETL process for a chosen data set.
Partial objectives include:

- Analysis of available information sources regarding machine learning and ETL processing.
- Choose a testing data set.
- Propose and develop a machine-learning model for ETL processing.
- Analyze and evaluate the suitability of the model based on selected metrics.

2.2 Methodology

The thesis will consist of two parts. The methodology of the theoretical part is based on analysing available literary and other information sources. The first chapter will introduce to Data Warehouse concept and define the position of the ETL process. Then will be given a short explanation of machine learning and its tasks. The next chapter will give deep learning concepts and MLPs definitions with possible tasks. At the end of the theoretical part will be talking about possible and chosen design environments.

In the practical part, the first chapter will aim at choosing of testing data set. Then Explanation of Data Analysis will be produced on the data set using additional Python packages such as NumPy, Pandas, etc. In the next chapter, a machine learning model will be proposed and developed using Python programming language, including additional Python packages specialized for data processing and machine learning (TensorFlow, Keras). The model will be trained and validated using chosen dataset. The Metric for the proposes of this model is cross-entropy.

3 Literature Review

3.1 Data Warehouse

Another part of process data is stored data. To store and manage data database is used. A database is a shared collection of logically related data and descriptions of that data, designed to meet the specific needs (of an organization). [4] Also, there are different types of architecture of databases.

Most cases in a company need two types of data:

1. Primitive data.
2. Derived data.

Primitive data is detailed data used to run the day-to-day operations of the company. Derived data has been summarized or otherwise calculated to meet the needs of the management of the company. [5] Because Derived data usually is a product of calculation it has a different approach to its life cycle opposite to Primitive data. Primitive data could be updated – derived data could be recalculated but not updated. Primitive data is up-to-date data – derived data often is historical data.

To sum up, it is hard to store primitive data and derived data in the same database. Primitive data and derived data are in different databases or different environments.

The figure below represented levels of the architecture. [5]

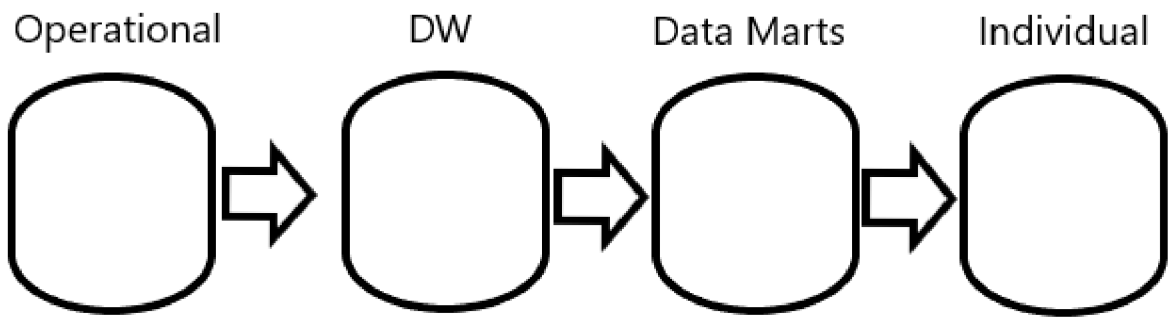


Fig. 1 Levels of the architecture. [5]

In general, operational is used for Primitive Data. Data at the operational level:

- Detailed.
- Day-to-day.
- Application proposes.
- High speed of manipulation of data.

The DW or Data Warehouse is used for Primitive and Derived Data. Data at the DW level:

- Primitive Historical data.
- Cannot be updated.
- Transformed for subject proposes.

Data Marts or departmental level are used for Derived Data. Data at the Data Marts level:

- Shaped for end-user.
- Specified for a department by requirements.

The individual level is used for Derived Data. Data at the individual level:

- Temporary.
- Used for analysis of jobs and tasks.

This concept of architecture allows us to separate maintenance, implementation of new solutions, and analysis of business insight in parallel with the day-to-day operation of a company.

Our ETL process is an essential part of linking the operational and DW levels. Thirst data are extracted from the source, second data are transformed in a place named Data Stage Area. In this area, data are prepared for the subject proposed. After that Data is loaded into the Data Warehouse.

As mentioned before ETL process is complicated. Very difficult to use simple technics of automation for this process.

In this thesis, I will take a corporate information factory architecture of database systems. Based on Fig. 1 and the architecture structure, it is a good point to design and implement machine learning to automate the ETL process. Since it will save cost and time for other significant tasks.

3.2 Machine learning

To start talking about machine learning model design, let's clarify what exactly is machine learning.

Machine learning can be defined as the process of teaching a machine to think like a human being to perform a particular task, without being explicitly programmed. [6]

As we know, machine learning allows for extracting knowledge from the data. It is possible because of using statistics, mathematics, and computer science fields in cooperating to imitate decision-making by the human brain. At the now moment machine learning are using in many spheres for example:

3.2.1 GPT and Genetic algorithm

The main model architecture was published in paper “Attention Is All You Need”. [8] And for the next five year that model was used in many and many machine learning models. Most known is ChatGPT. ChatGPT is a large language model that can be used in a variety of applications. It could be used in chatbots for customer service, allowing companies to quickly respond to inquiries and provide helpful information to customers. ChatGPT could also be integrated into virtual assistants, such as Siri or Alexa, to improve their ability to understand and respond to user requests. In the field of education, ChatGPT could be used as a tutor, providing personalized learning experiences to students. It could also be used in the field of healthcare, helping doctors and nurses to diagnose and treat patients by analyzing medical data.

3.2.2 Detection Algorithms

Several machine-learning algorithms can be used in cancer detection to analyze patient data and identify patterns or abnormalities that may indicate the presence of cancer. Some of the commonly used algorithms include:

- Support Vector Machines (SVMs):
  SVMs are a type of supervised learning algorithm that can be used for classification tasks, such as distinguishing between cancer patients and healthy individuals. SVMs work by finding the optimal hyperplane that separates the two classes of data in a high-dimensional feature space. In the case of cancer detection, the features could be genetic or proteomic data, and the SVM can identify patterns in the data that can help distinguish between cancer and non-cancer samples.
- Random Forest:
  Random Forest is an ensemble learning algorithm that combines multiple decision trees to create a more accurate and stable model. It can be used to identify important features or biomarkers that are associated with cancer and to build

predictive models for patient outcomes. Random Forest can also be used to estimate the importance of each feature in the data and provide a ranking of the most informative biomarkers for cancer detection.

- Neural Networks:
  Neural Networks are a type of machine learning algorithm that are designed to mimic the way the human brain works. They can be used to analyze medical images, such as CT or MRI scans, to detect tumors or other abnormalities. Neural Networks consist of layers of interconnected nodes, which can learn to recognize patterns in the data through a process of training. In the context of cancer detection, the neural network can be trained on a large dataset of medical images, and then used to predict whether a new image contains cancer or not.
- K-Nearest Neighbors (KNN):
  KNN is a type of instance-based learning algorithm that can be used for both classification and regression tasks. It can be used to group patients with similar characteristics and identify subgroups that are more likely to develop cancer. In KNN, each data point is assigned to the class of its k-nearest neighbors, based on a distance metric such as Euclidean distance. For example, KNN can be used to group patients based on their genetic or proteomic profiles and identify subgroups that have a higher risk of developing cancer.
- Decision Trees:
  Decision Trees are a type of machine learning algorithm that can be used to generate rules for predicting the presence or absence of cancer based on patient data. Decision Trees work by recursively partitioning the data into subsets based on the most informative features, until a stopping criterion is met. The resulting tree can be used to classify new data based on the rules learned during training. For example, a decision tree can be trained on a dataset of patient characteristics and biomarker data to predict whether a patient is at risk of developing cancer or not.

3.2.3 Remote Sensing and Classification algorithms

1. Machine learning algorithms can be used in remote sensing to analyze large amounts of satellite imagery and other sensor data, and to extract meaningful information about the environment. This information can be used to monitor environmental changes, identify

areas of concern, and develop solutions to environmental problems. Here are some ways in which machine learning can be used in remote sensing for environmental solutions:

- One of the main applications of machine learning is land cover classification, where algorithms are trained to classify different types of land cover based on satellite imagery. This information is useful for monitoring changes in land cover over time and identifying areas at risk of deforestation or urbanization. Commonly used algorithms for land cover classification include Random Forest, Support Vector Machines, and Convolutional Neural Networks.
- Another application of machine learning in remote sensing is vegetation monitoring, which involves analyzing spectral data from satellite sensors to identify areas of stress or disease in vegetation. This is done by detecting changes in the amount of chlorophyll and other pigments in vegetation, which can be an indicator of health. Machine learning algorithms, such as Decision Trees and Neural Networks, are commonly used in this field.
- Machine learning also plays a critical role in climate change modelling, where predictive models are built to forecast changes in temperature, precipitation, sea level, and other variables, and to develop strategies for adapting to these changes. Commonly used models include Artificial Neural Networks, Support Vector Regression, and Random Forest Regression.
- Air pollution monitoring is another area where machine learning algorithms are used to identify sources of pollution, such as factories or transportation networks, and to develop policies and interventions to reduce air pollution and improve air quality. This is done by analyzing data from air quality sensors and satellite imagery. Commonly used algorithms in this field include Decision Trees and Neural Networks.
- Finally, machine learning algorithms are also used in disaster response, where they can analyze satellite imagery and identify areas affected by natural disasters, such as floods or wildfires. This information can be used to direct emergency response efforts and assess the extent of the damage. Algorithms commonly used in this field include Convolutional Neural Networks and Deep Learning models.

One example of the implementation of machine learning algorithms in environmental science is the use of a Random Forest algorithm to classify land cover types in a study of the Amazon rainforest. The study used satellite imagery to classify different types of vegetation and found that the Random Forest algorithm was able to accurately classify land cover types with an accuracy of 93.8%.

Another example is the use of Artificial Neural Networks to develop predictive models of sea level rise in a study of the Chesapeake Bay. The study used data from satellite sensors and historical sea level data to develop a model that could predict future sea level rise with an accuracy of 86%.

In general, machine learning can be divided into three types of approaches to design:

1. Supervised Learning.
2. Unsupervised Learning.
3. Reinforcement Learning.

Supervised Learning is one of the most used machine learning paradigm. Supervised Learning or supervised learning algorithms required that users provide both input and output data. In this problem, the task T is to learn a mapping f from inputs $x \in X$ to outputs $y \in Y$. The input x are also called the features, covariates, or predictors; this is often a fixed dimensional vector of numbers, such as the height and weight of a person, or the pixels in an image. [7] For example, we would like to know which type of transaction going through data storage area to DW to do that user must provide the transaction as input and label or name of a type of transaction that gives the possibility for the machine to map or learn the relation between input and output. Generally, a supervised machine learning model requires humans to support loading training sets with pre-defined labels.

We have two major supervised machine-learning problems to solve:

- Regression.
- Classification.

Classification problems define the goal to predict a class label. Also, the predicted class label should be from previously given labels. Classification problems also dividing into two types as well supervised machine-learning problems:

- Binary classification.
- Multiclass classification.

Binary classification is used for cases when the output should have only two possible labels. For example, email sorting to spam or not spam, survey predictions, etc. Multiclass classification is used for cases when the output must be more than two classes. That might be the classification of emails by NDA.

Regression problems define the goal to predict the continuous number. For example, we would like to predict the price of a flat on market or predict the price of electricity.

Unsupervised learning is another big part of machine learning algorithms. Opposite to supervised learning, we more often don't know what output should be. In this case, the machine learning model takes an input and a task to extract some insights from it. Unsupervised learning algorithms may be divided into several groups:

- Clustering.
- Association.
- Transformations.
- Etc.

Clustering algorithms are used on uncategorized data to find structure or pattern in it. Usually, the user could control how many clusters (groups) the algorithm should find.

Association algorithms are used to find or investigate relationships between variables.

- Transformations.

Unsupervised transformations of a dataset are algorithms that create a new representation of the data which might be easier for humans or other machine learning algorithms to understand compared to the original representation of the data. [9]

Reinforcement learning (RL) algorithms involve the strategy of learning via interacting (sequences of actions, observations, and rewards) with the environment. [10]

3.3 Deep learning

Deep learning is another big part of machine learning. Deep learning in general tries to imitate the working of biological neurons in machine learning. This approach leads to the artificial neural network (ANNs). ANNs are a core of deep learning, as well as very powerful and well enough for the thesis proposed architecture. To clarify the ANNs need to get a few historical notes.

ANNs have been around for quite a while: they were first introduced back in 1943 by the neurophysiologist Warren McCulloch and the mathematician Walter Pitts. [11] They used the animal brain as an example of working neurons then they perform very complex computations using propositional logic to represent the first artificial neural network architecture.

An artificial neuron is an essential part of an artificial neural network. Also, an artificial neuron is a mathematical representation of a biological neuron.

McCulloch and Pitts proposed a very simple model of the biological neuron, which later became known as an artificial neuron: it has one or more binary (on/off) inputs and one binary output. [11] Fig. 2 represented the ANNs performing simple logical computations:

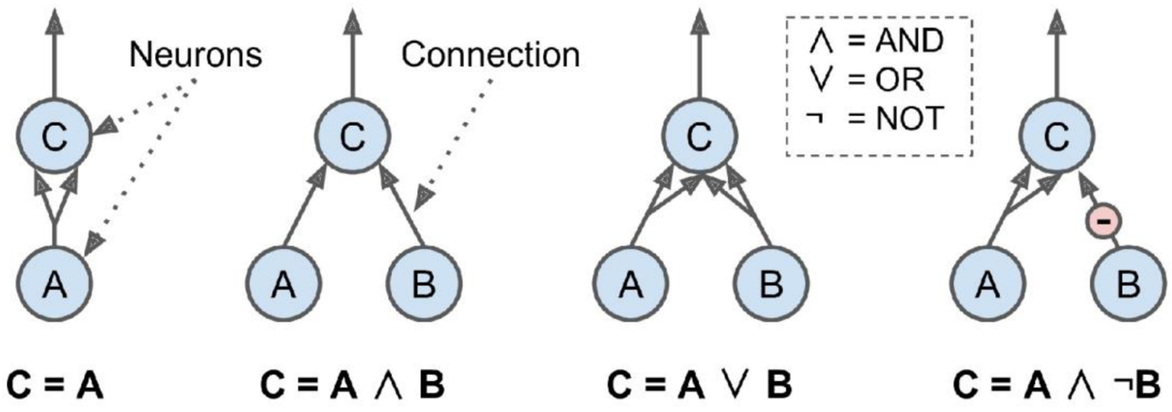


Fig. 2 ANNs performing simple logical computations. [11]

That was the first primitive system of ANNs. By the simple logic with „one“ and „zero“. But there should be notes that it is still an artificial network and we could have hundreds of layers of artificial neurons to solve complex and difficult logic expressions.

The next step in ANNs was perceptron. The Perceptron is one of the simplest ANN architectures, invented in 1957 by Frank Rosenblatt. It is based on a slightly different artificial neuron called a threshold logic unit (TLU), or sometimes a linear threshold unit (LTU). [11]

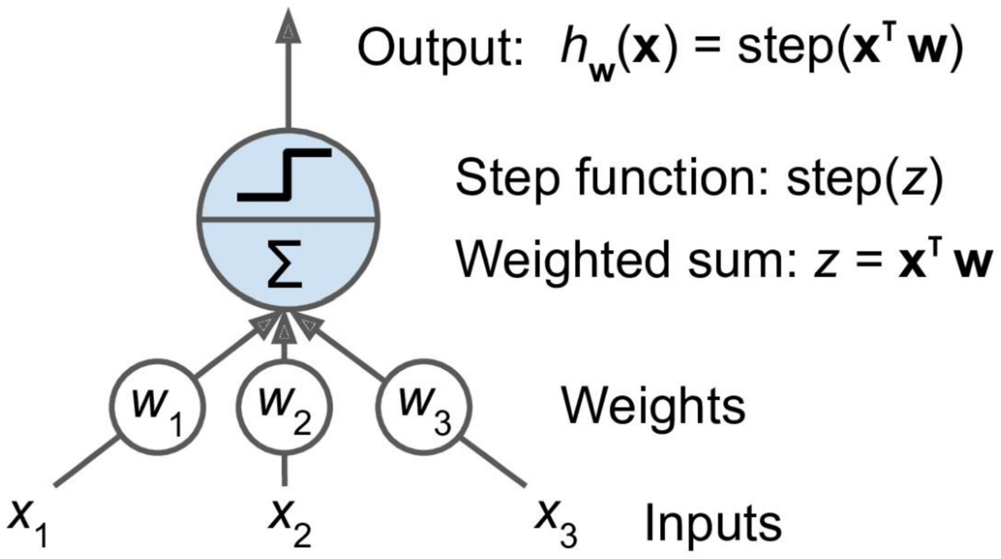

Output: $h_{\mathbf{w}}(\mathbf{x}) = \text{step}(\mathbf{x}^T \mathbf{w})$
Step function: step(z)
Weighted sum: $z = \mathbf{x}^T \mathbf{w}$
Σ
w_1
w_2
w_3
Weights
x_1
x_2
x_3
Inputs

Fig. 3 Threshold logic unit: an artificial neuron that computes a weighted sum of its inputs and then applies a step function. [11]

On Fig. 3 represented a simple perceptron. The main difference with artificial neurons is that we have output from 0 to 1.

The algorithms behind that are going by following steps:

1. We have an inputs matrix of features x1, x2, x3 and the corresponding weights (coefficient)
2. TLU calculates the weighted sum by the following rule:

$$z = (x1 * w1 + x2 * w2 + x3 * w3 + xn * wn) \quad (1)$$

3. Then applies the step function

$$H_w(x) = step(z), where\ z = x^T w \quad (2)$$

Opposite to the previous example of imitating neurons, using one TLU we can implement binary classification, for example, logical regression or SVM classifier. Training TLU means finding correct or fit w0, w1, and w2.

A Perceptron is simply composed of a single layer of TLUs, with each TLU connected to all the inputs. When all the neurons in a layer are connected to every neuron in the previous layer (i.e., its input neurons), the layer is called a fully connected layer, or a dense layer. [11]

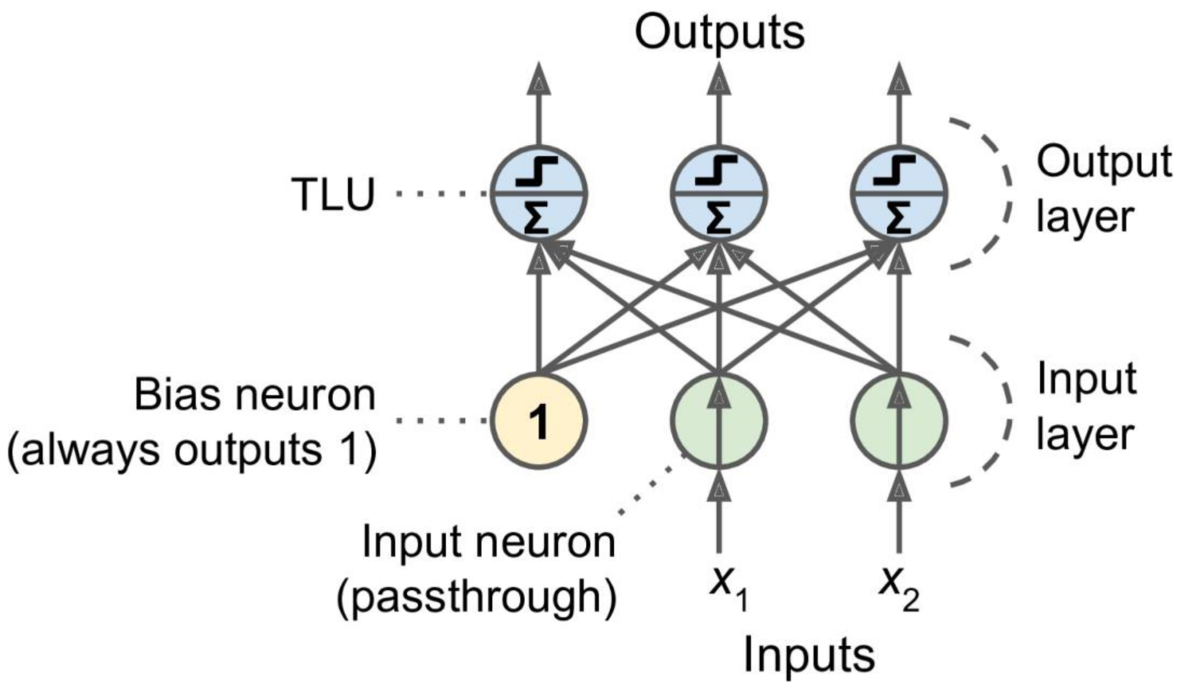


Fig. 4 Architecture of a Perceptron with two input neurons, one bias neuron, and three output neurons. [11]

Two input neurons have *x1 and x2 matrices* of input feature one row per instance and one column per feature. These two inputs are allocated in the input layer. Then these inputs are fed to the TLU with their weights *matrix W*, also there we have *bias vector b* that represents all the connection weights between the bias neuron and the artificial neurons. It has one bias term per artificial neuron. And in general, we have an activated function, for the case in Fig. 4 it is a step function. Using the mathematics and explanation above we could say that output could be calculated by the following function:

$$h_{W,b}(X) = f(XW + b) \tag{3}$$

As mentioned above, training of perceptron is finding of weights. Using the previous research on perceptron, we can use formulate the perceptron learning rule:

$$w_{i,j}^{(next\ step)} = w_{i,j} + \eta(y_j - \hat{y}_j)x_i \tag{4}$$

Where,

- w is the connection weight between the ith input neuron and the jth output neuron.
- x is the ith input value of the current training instance.
  $\hat{y}_j$ is the output of the jth output neuron for the current training instance.
- y_j is the target output of the jth output neuron for the current training instance.
- η learning rate [9]

Unfortunately, the architecture in Fig. 4 cannot solve the basic logic operation XOR (exclusive OR). After research and innovation, it was decided to use multiple perceptrons, and as a result, this ANN was named Multilayer Perceptron (MLP).

3.3.1 Multilayer Perceptron

The idea of multilayer perceptron is similar to previous architecture. Multilayer perceptron has three layers.

First, the input layer or passthrough.

Second, the hidden layer this layer consists of one or more layers of TLUs.

Third, the output layer consists of the TLUs layer as well.

There should be notes that each layer except the final has a bias neuron. Sometimes layers that are close to the passthrough or input layer are usually called lower layers. And the layers that are close to the output layer are usually called upper layers.

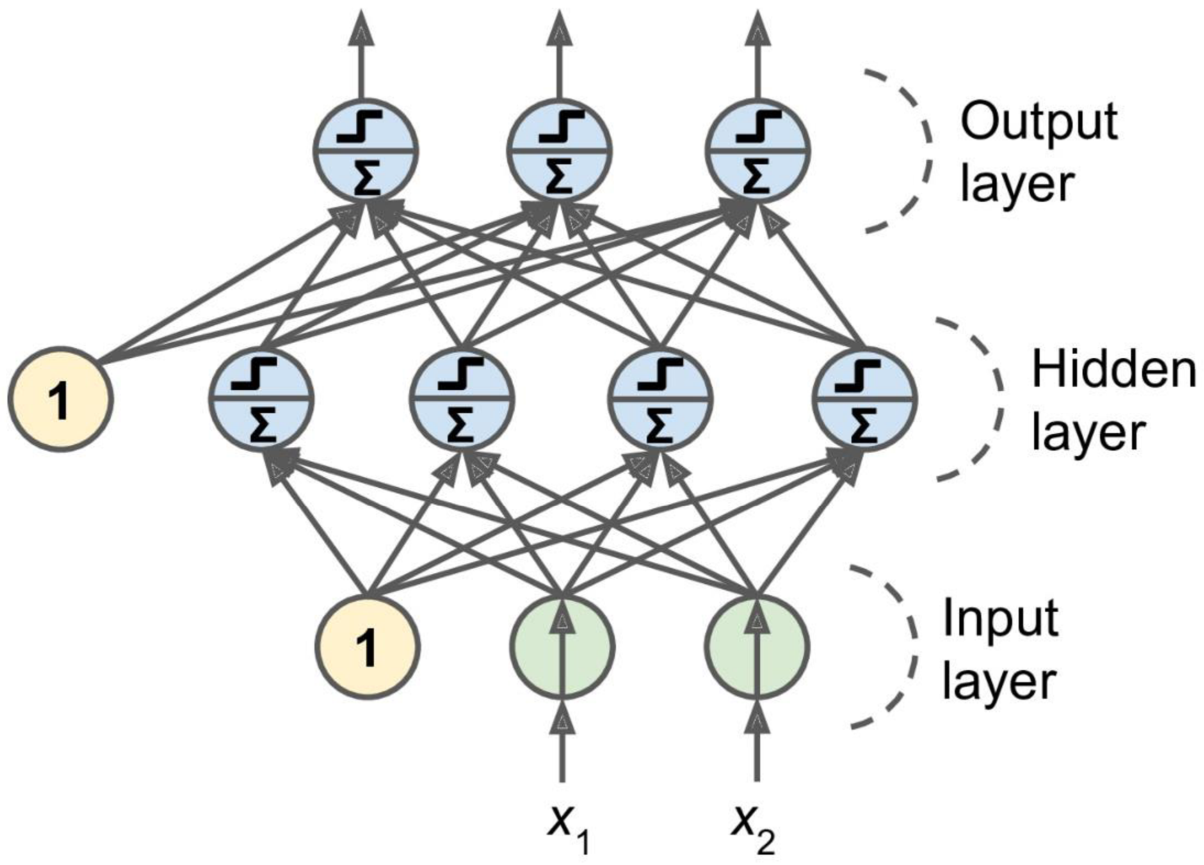


Fig. 5 Architecture of a Multilayer Perceptron with two inputs, one hidden layer of four neurons, and three output neurons (the bias neurons are shown here, but usually they are implicit) [11]

For a long time was a problem to train MLPs. In 1986 David Rumelhart, Geoffrey Hinton, and Ronald Williams represent their backpropagation training algorithm. This algorithm focuses on the following steps:

- It processes one group at once (the size of the group might be 32 examples). And it goes through the whole dataset a few times. Each pass is named epoch.
- Then each group passes to the input layer, then the output computer and goes to the next layer, and so on, until the output layer. This is named forward pass. Should be noted that a backward pass is impossible without a forward pass. Each example from the group should be passed to the input layer.
- Next step is using a loss function to calculate the error, usually loss function aims to take the desired output and compare it to the actual output, as a result, it returns a measure of error.
- After that, using the chain rule algorithm computes the effect of each output on the error.
- Then it computes how much each output effect the error, again using the chain rule it goes down from the upper layer to the lower layer until it reaches the input layer. The pass measures the gradient error through all layers.
- Final step is using the Gradient descent step to redefine the weight of each connection between layers.

Also, the researchers modify the previous model of MLP by changing the activation function from the step function to the logic one named sigmoid.

There are several types of activation functions:

- ReLu

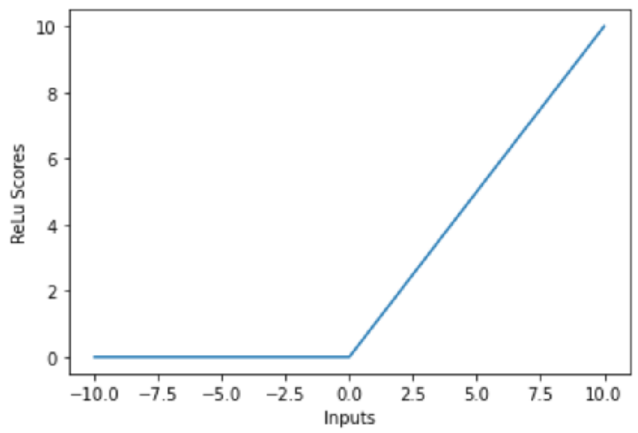


Fig. 6 ReLu activation function, source: author

- Softmax

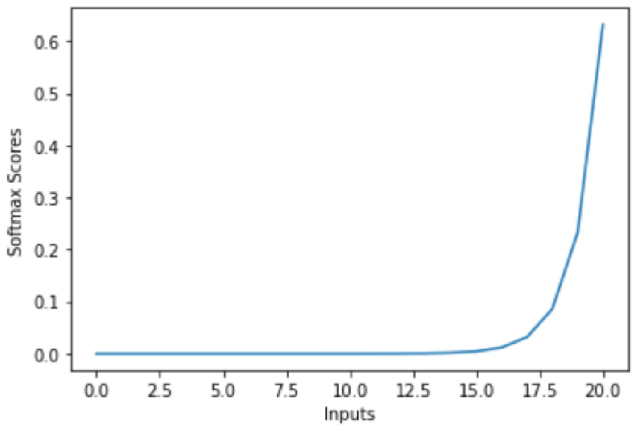


Fig. 7 Softmax activation function, source: author

3.3.2 Classification MLPs

The general classification problem in terms of MLPs is divided into three groups:

1. Binary classification.

Binary classification is a type of classification problem where the objective is to classify data into two distinct categories or classes. In the context of MLPs, this involves training a neural network to learn a decision boundary that separates the two classes. The output layer of the MLP will typically have a single neuron that outputs a probability value between 0 and 1, which can be interpreted as the likelihood of the input belonging to one of the two classes.

Advantages of MLPs for binary classification:

- MLPs are flexible and can learn complex decision boundaries, which makes them suitable for a wide range of binary classification problems.
- MLPs can handle a large number of input features, which is important for many real-world applications.
- MLPs can be trained using backpropagation, which is an efficient optimization algorithm for neural networks.

Disadvantages of MLPs for binary classification:

- MLPs can be prone to overfitting if the network architecture or training process is not carefully designed.
- MLPs can be computationally expensive to train, especially for large datasets or complex network architectures.
- MLPs can be sensitive to the choice of hyperparameters, such as the learning rate or the number of hidden layers.

2. Multilabel binary classification.

Multilabel binary classification is a type of classification problem where each input can be assigned to one or more of a set of possible labels. In the context of MLPs, this involves training a neural network to output a binary vector of size N, where N is the number of possible labels, and each element of the vector represents whether the input belongs to the corresponding label or not.

Advantages of MLPs for multilabel binary classification:

- MLPs can handle a large number of possible labels, which is important for many real-world applications.
- MLPs can learn complex relationships between the input and the labels, which can improve the accuracy of the predictions.
- MLPs can be trained using various loss functions, such as binary cross-entropy or sigmoid cross-entropy, which can be tailored to the specific problem at hand.

Disadvantages of MLPs for multilabel binary classification:

- MLPs can be prone to overfitting if the network architecture or training process is not carefully designed.
- MLPs can be computationally expensive to train, especially for large datasets or complex network architectures.
- MLPs can be sensitive to the choice of hyperparameters, such as the learning rate or the number of hidden layers.

3. Multiclass classification.

Multiclass classification is a type of classification problem where the objective is to classify data into more than two distinct categories or classes. In the context of MLPs, this involves training a neural network to output a probability vector of size N, where N is the

number of possible classes, and each element of the vector represents the likelihood of the input belonging to the corresponding class.
Advantages of MLPs for multiclass classification:

- MLPs can handle a large number of possible classes, which is important for many real-world applications.
- MLPs can learn complex decision boundaries that can separate the input into multiple classes.
- MLPs can be trained using various activation functions, such as softmax, which can be tailored to the specific problem at hand.

Disadvantages of MLPs for multiclass classification:

- MLPs can be prone to overfitting if the network architecture or training process is not carefully designed.
- MLPs can be computationally expensive to train, especially for large datasets or complex network architectures.
- MLPs can be sensitive to the choice of hyperparameters, such as the learning rate or the number of hidden layers.

3.3.3 Loss function

The loss function is a function that tells how the algorithm models the dataset.
If predictions are mismatched, the loss function will output high numbers.
The loss function for all these classification MPLs is the cross-entropy function.

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverge from the actual label. So, predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0. [12]

3.4 Environmental Solutions

Python and R are two of the most widely used programming languages in the field of machine learning. Both languages have their own advantages and disadvantages, making them suitable for different use cases. In this diploma paragraph, we will discuss what Python and R are, their pros and cons for working with machine learning, and compare the two languages.

Python is a high-level, interpreted programming language that was created in 1991. It is known for its simplicity and ease of use, making it a popular choice among beginners and experienced programmers alike. Python has a vast library support, including libraries such as Scikit-Learn, TensorFlow, PyTorch, and Keras, which make it an ideal language for machine learning. Python's extensive library support provides developers with a range of tools to build complex machine learning models with ease.

One of the significant advantages of using Python for machine learning is its versatility. Python can be used for various purposes, such as web development, scientific computing, and data analysis, making it a go-to language for many professionals. Moreover, Python's extensive community support provides users with an abundance of resources, tutorials, and forums to help them solve any issues they may encounter.

However, Python also has some disadvantages when it comes to machine learning. One of the drawbacks of Python is its performance. Although Python is a high-level language, its interpreted nature can make it slower than other compiled languages, such as C++ and Java. However, this can be mitigated by using libraries like Numpy and Pandas, which provide optimized algorithms for data manipulation.

R is another popular programming language used in machine learning. It is a language specifically designed for data analysis and visualization, making it an excellent choice for statisticians and data analysts. R has a vast library support, especially for statistical analysis and graphing, which are critical for data analysis. One of the significant advantages of R is its ability to handle large datasets and perform complex statistical analyses.

When compared to Python, R has some disadvantages. One of the significant drawbacks of R is its syntax, which can be difficult to learn and understand, especially for beginners. Moreover, R's community support is not as extensive as Python's, making it challenging to find support and resources.

In terms of machine learning, both Python and R have their own strengths and weaknesses. Python's machine learning libraries provide more advanced features, such as deep learning and natural language processing, making it a better choice for complex machine learning projects. R, on the other hand, is better suited for statistical analysis and graphing, making it an excellent choice for data analysts and statisticians.

Another advantage of using Python for machine learning is the availability of a range of tools and frameworks, such as TensorFlow, PyTorch, and Keras. These tools enable developers to build and deploy deep learning models with ease. Python also has a larger user community than R, making it easier to find help and support when encountering issues.

On the other hand, R's strengths lie in its ability to handle large datasets and perform complex statistical analyses. R has an extensive library of statistical tools, making it the preferred choice for data analysis and visualization tasks. The ggplot2 package, for instance, provides an easy-to-use interface for producing high-quality plots and visualizations.

However, R does have some disadvantages. The language can be difficult to learn, and it is not as versatile as Python. Also, R's syntax can be challenging to read and understand for beginners. The community support for R is not as extensive as Python's, which can make it difficult to find resources and support.

When it comes to choosing between Python and R for machine learning, it ultimately comes down to the specific requirements of the project. Python is the go-to language for deep learning and natural language processing tasks, while R is better suited for statistical analysis and visualization tasks.

3.4.1 Python language

In 1990 Guido van Rossum, a Dutch mathematician, at the Centrum Wiskunde & Informatica in the Netherlands developed and released Python. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.

The first version of Python was Python 0.9.0 in 02.1991 [13]

With the aim of deep learning python has the following advantages:

- Short and readable code that helps to consume the time between iterations.
- Large community that helps to not to make a wheel.
- Open-source programming language.
- Allow collaborative codding.
- Scripting language.

3.4.2 R language

R is an open-source implementation of the S language created and developed at Bell Labs. S is also the basis of the commercial statistics program S-PLUS, but R has eclipsed S-PLUS in popularity. [14] R was first released and developed in the early 1990s by Robert Gentleman and Ross Ihaka from the University of Auckland. But it was not the first version. It was closely modelled on the S Language. The best sentence that explains the essentials of R language is a language written by statisticians for statisticians.

Advantages of R language with a focus on deep learning and machine learning:

- Open-source programming language.
- Support all operating systems.
- Can be collaborated with many other languages.
- Large community.
- Official GUI RStudio.

Both languages have advantages and disadvantages. This thesis is aim to MLPs deep learning model that is why I going to use Python.

4 Practical Part

Before starting the practical part, lets summaries important points from theoretical part:

- The main goal of this work is developing and evaluating the machine learning model.
- I will be using the deep learning model for the classification of transaction trading data.
- The process of classification of transaction trading data take place in ETL stage of Data Warehouse.
- The machine learning model that will be using in this paper is neural network multiclassification model.
- A loss function, cross-entropy will be using.
- To evaluating of machine learning model I will be using Precision, Recall, F1 score, confusion matrix and area under curce of Receiver Operating Characteristic (AUC ROC).

  Need to note that machine learning process mainly needs hardware for mathematical calculation. That need is depending on the complexity of the algorithm. Hardware that will be using in this paper represented below:
- Processor: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.81 GHz
- RAM: 8 GB
- System: Windows 10 Home, version: 21H2
- Hard Driver: Samsung mzvlw128hegr-000h1 SSD 128 GB

For this thesis, I will use Python version 3.10.2 that are more reliable.

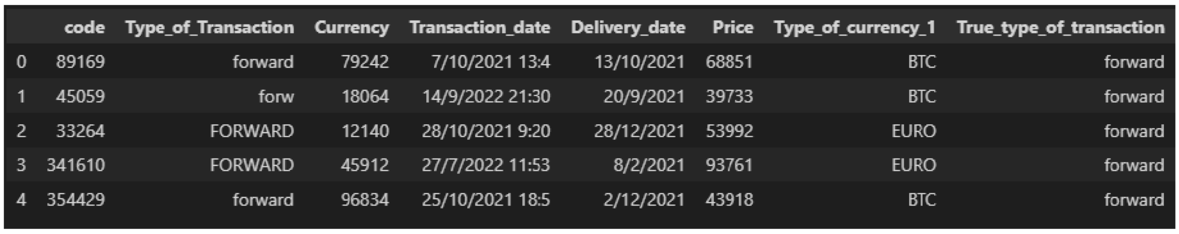
4.1 Data set

Transaction data are sensitive data and protected by companies. Due to that fact, I will produce my datasets. Datasets will cover the following patterns:

- Type of transaction: SWAP, FORWARD, FUTURE.
- Type of currency: EUR, BTC.
- Different Datetime format: YYYY-MM-DD, DD/MM/YYYY.
- Based on the type of transaction different numbers of fields per row.

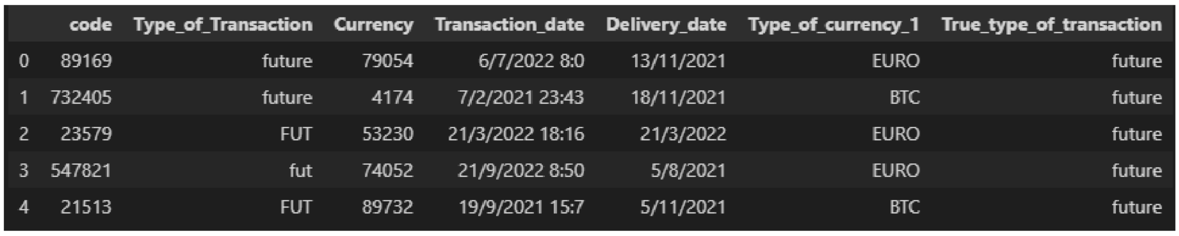
- Based on the type of transaction different order of fields.

The dataset will be made in Python using libraries such as random and csv. The following figures represent the results of script production.



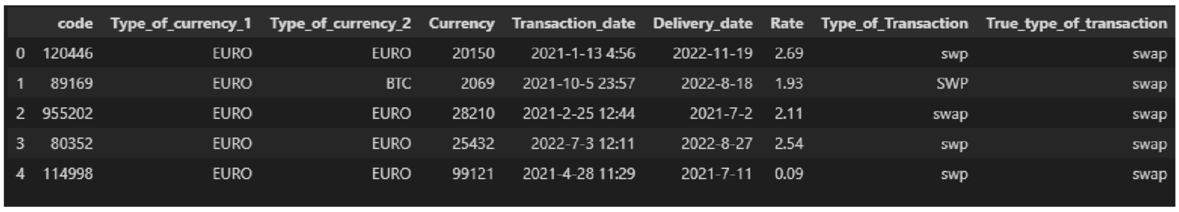
	code	Type_of_Transaction	Currency	Transaction_date	Delivery_date	Price	Type_of_currency_1	True_type_of_transaction
0	89169	forward	79242	7/10/2021 13:4	13/10/2021	68851	BTC	forward
1	45059	forw	18064	14/9/2022 21:30	20/9/2021	39733	BTC	forward
2	33264	FORWARD	12140	28/10/2021 9:20	28/12/2021	53992	EURO	forward
3	341610	FORWARD	45912	27/7/2022 11:53	8/2/2021	93761	EURO	forward
4	354429	forward	96834	25/10/2021 18:5	2/12/2021	43918	BTC	forward

Fig. 8 Dataframe of the forward dataset, source: author



	code	Type_of_Transaction	Currency	Transaction_date	Delivery_date	Type_of_currency_1	True_type_of_transaction
0	89169	future	79054	6/7/2022 8:0	13/11/2021	EURO	future
1	732405	future	4174	7/2/2021 23:43	18/11/2021	BTC	future
2	23579	FUT	53230	21/3/2022 18:16	21/3/2022	EURO	future
3	547821	fut	74052	21/9/2022 8:50	5/8/2021	EURO	future
4	21513	FUT	89732	19/9/2021 15:7	5/11/2021	BTC	future

Fig. 9 Dataframe of the future dataset, source: author



	code	Type_of_currency_1	Type_of_currency_2	Currency	Transaction_date	Delivery_date	Rate	Type_of_Transaction	True_type_of_transaction
0	120446	EURO	EURO	20150	2021-1-13 4:56	2022-11-19	2.69	swp	swap
1	89169	EURO	BTC	2069	2021-10-5 23:57	2022-8-18	1.93	SWP	swap
2	955202	EURO	EURO	28210	2021-2-25 12:44	2021-7-2	2.11	swap	swap
3	80352	EURO	EURO	25432	2022-7-3 12:11	2022-8-27	2.54	swp	swap
4	114998	EURO	EURO	99121	2021-4-28 11:29	2021-7-11	0.09	swp	swap

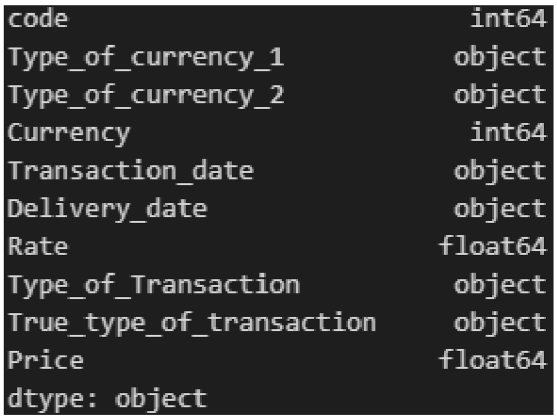
Fig. 10 Dataframe of swap dataset, source: author

4.2 Exploration data analysis

Before starting any machine learning project, exploration data analysis must be added. To perform the analysis, Pandas and c libraries will be used.

The pandas is a high-performance open source library for data analysis in Python developed by Wes McKinney in 2008. [15]

Concatenate the datasets into one dataset [Appendix 1.] Then using function dtypes got the data types in the dataset.



```
code                          int64
Type_of_currency_1           object
Type_of_currency_2           object
Currency                      int64
Transaction_date             object
Delivery_date                object
Rate                        float64
Type_of_Transaction          object
True_type_of_transaction     object
Price                       float64
dtype: object
```

Fig. 11 Result of the dtype function, source: author

In the dataset, I have four numerical variables and six categorical variables. A machine learning task that will be represented in this thesis is the classification task.

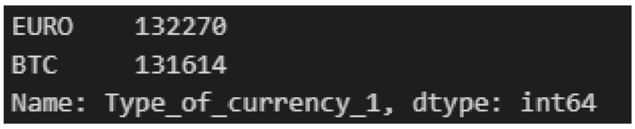
Using the following column, I will design the machine learning model to predict the type of transaction based on the input features.

The next point is going through all variables to get more information about the data and producing preprocessing of data.

In the database system, the code column will be used as a primary key. This leads to the requirement for uniqueness. After the concatenation of datasets, the code column consists of duplicates. Using Pandas the following numbers:

Missing value in code column 36113 and the whole dataset is 299997 numbers of rows. Due to the small number of duplicates, I will drop them from the dataset. Machine learning requires only numerical data as input.

Type of currency 1 and type of currency 2 are binary categorical variables that could be BTC or EURO.



```
EURO    132270
BTC     131614
Name: Type_of_currency_1, dtype: int64
```

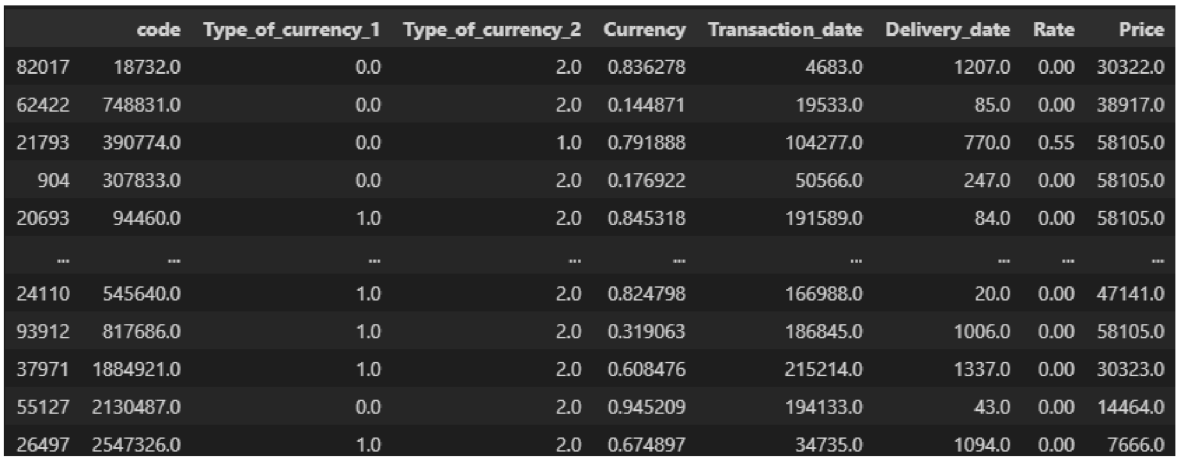
Fig. 12 Count of variable in Type of currency 1 column, source: author

Column Type of currency 2 appears only in swap transactions. Opposite to the code approach, I will replace NaN values with the string None. That will help in the label encoding of categorical data for the machine learning model. Then using the scikit-learn library and label encoding technique, I will encode the Type of currency 1 and type of currency 2 columns. [Appendix 2.]

The currency, Rate and Price columns are numerical. To increase the speed of performance of the machine learning model these columns will encoding using MinMaxScaler from the scikit-learn library. The missing values of the Rate and Price columns were replaced with 0. [Appendix 3.]

The categorical variables as Transaction date, Delivery date and target column True type of transaction will label encoding.

The split training dataset that will be used in the learning of the model is represented below.



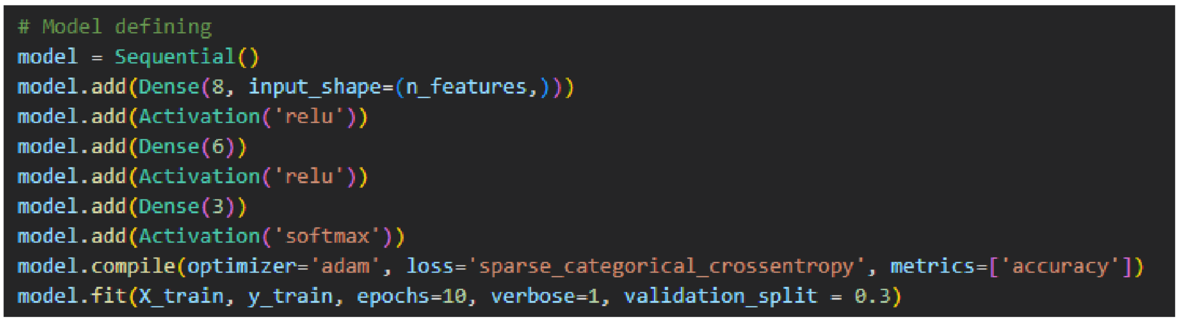
	code	Type_of_currency_1	Type_of_currency_2	Currency	Transaction_date	Delivery_date	Rate	Price
82017	18732.0	0.0	2.0	0.836278	4683.0	1207.0	0.00	30322.0
62422	748831.0	0.0	2.0	0.144871	19533.0	85.0	0.00	38917.0
21793	390774.0	0.0	1.0	0.791888	104277.0	770.0	0.55	58105.0
904	307833.0	0.0	2.0	0.176922	50566.0	247.0	0.00	58105.0
20693	94460.0	1.0	2.0	0.845318	191589.0	84.0	0.00	58105.0
...	...	...	...	...	...	...	...	...
24110	545640.0	1.0	2.0	0.824798	166988.0	20.0	0.00	47141.0
93912	817686.0	1.0	2.0	0.319063	186845.0	1006.0	0.00	58105.0
37971	1884921.0	1.0	2.0	0.608476	215214.0	1337.0	0.00	30323.0
55127	2130487.0	0.0	2.0	0.945209	194133.0	43.0	0.00	14464.0
26497	2547326.0	1.0	2.0	0.674897	34735.0	1094.0	0.00	7666.0

Fig. 13 Training dataset, source: author

4.3 Machine learning model design

The first step is loading the required libraries. For the MLPs model, I will use TensorFlow and Keras. Scikit-learn will be used to split the data set to train and test. To more understandable of context of learning process of neural network need be defined:

- Epoch – is when all dataset is passed backward or forward through machine learning model at once.
- Batch – is a portion of dataset, that used to decrease memory cost of learning process.



```
# Model defining
model = Sequential()
model.add(Dense(8, input_shape=(n_features,)))
model.add(Activation('relu'))
model.add(Dense(6))
model.add(Activation('relu'))
model.add(Dense(3))
model.add(Activation('softmax'))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, verbose=1, validation_split = 0.3)
```

Fig. 14 MLPs classification model defining, source: author

The shape of the input is equal to the number of columns and equal to 8.

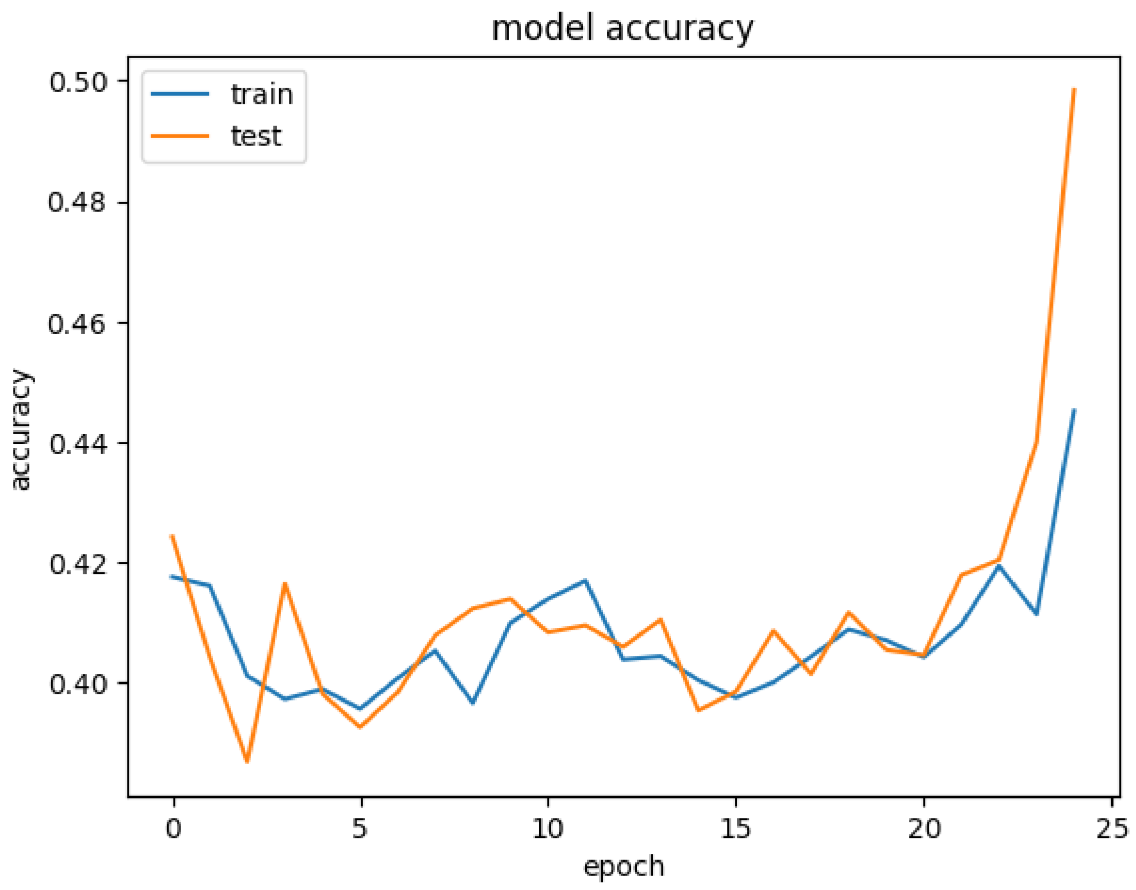


Fig. 15 Model accuracy depends on the number of epochs, source: author

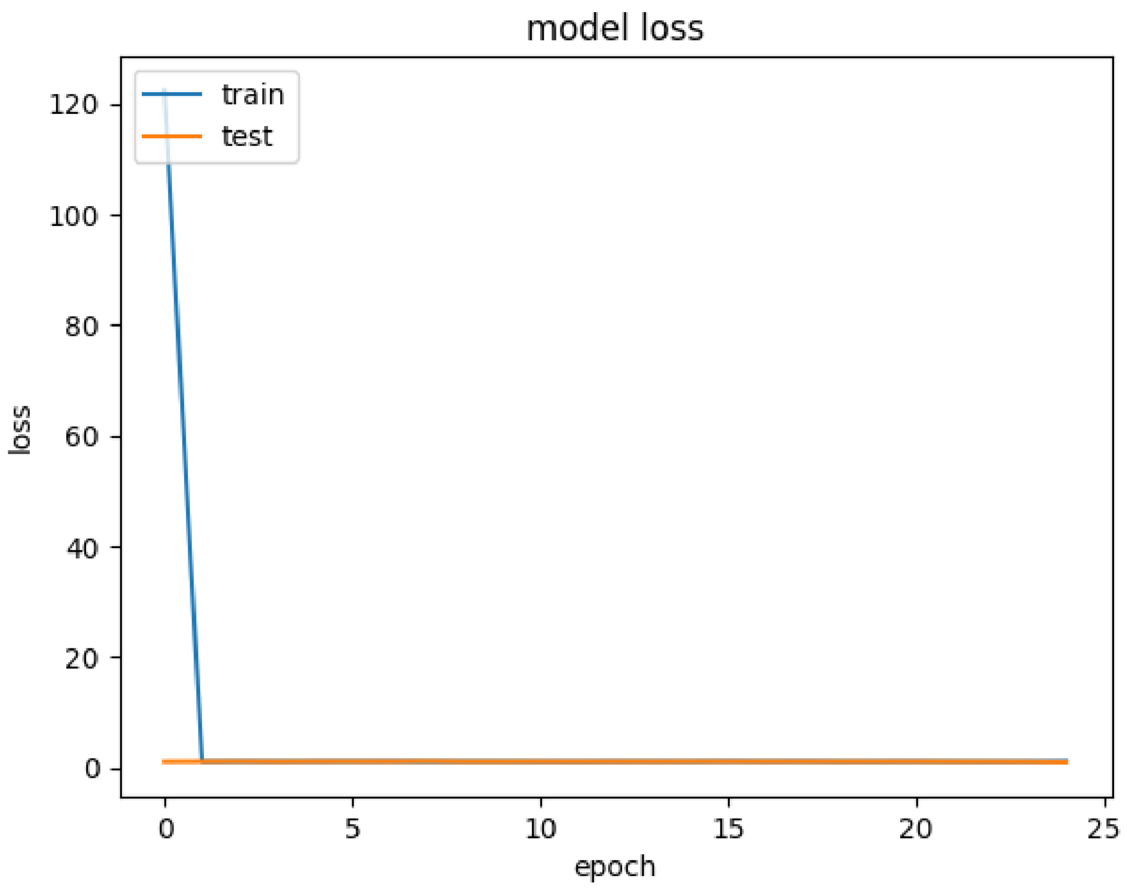


Fig. 16 Loss function depends on a number of epochs, source: author

The result of the machine learning training process is not accepted. The resulting accuracy of the model is equal to 44.52%. The accuracy of the model seems like learning model, but if we have a look at the loss function graph it completely disaster. That behaviour of loss function means incorrect decision-making of hyperparameters of the machine learning model.

4.3.1 Tuning of Hyperparameters

Hyperparameters are user-defined settings that are not learned during the training process of a machine learning model, but rather are set by the user before training begins. These settings can significantly impact the performance of the model and therefore, choosing appropriate hyperparameters is crucial for achieving optimal results. In this explanation, we will focus on hyperparameters for the Multilayer Perceptron (MLP) deep learning model.

The MLP is a type of neural network that consists of multiple layers of interconnected neurons. It is widely used for classification and regression tasks. Here are some of the key hyperparameters that can be tuned in an MLP model:

- Number of hidden layers: An MLP model can have one or more hidden layers, with each layer containing a certain number of neurons. The number of hidden layers and neurons in each layer are key hyperparameters that can affect the complexity and learning capacity of the model.
- Activation function: An activation function is used to introduce non-linearity to the output of each neuron. Commonly used activation functions include sigmoid, ReLU, and tanh. Choosing an appropriate activation function is important for ensuring that the model can learn complex relationships between the input and output.
- Learning rate: The learning rate is a hyperparameter that controls how quickly the model adjusts its weights during training. A higher learning rate can cause the model to converge faster, but it can also cause instability and make the model prone to overshooting the optimal solution.
- Batch size: The batch size is the number of training examples used in each iteration of the training process. A larger batch size can speed up training, but it can also lead to overfitting, where the model performs well on the training data but poorly on new, unseen data.
- Regularization: Regularization techniques such as L1, L2, and dropout can be used to prevent overfitting in the model. These hyperparameters control the amount of regularization applied during training.
- Optimization algorithm: The choice of the optimization algorithm used during training can also affect the performance of the model. Commonly used algorithms include Stochastic Gradient Descent (SGD), Adam, and Adagrad.

4.3.1.1 Activation function

One of the hyperparameters for the MLPs model is the activation function. As I mentioned above, the activation function for the output layer should be softmax to have probability output. In this case, I will replace the ReLU activation function in the hidden layer with the ELU activation function.

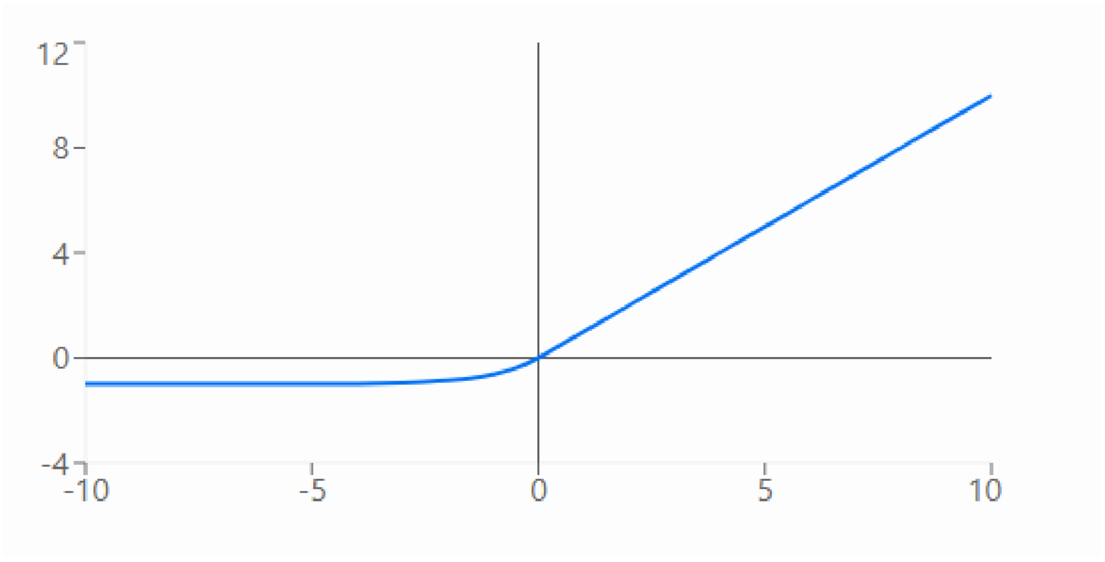


Fig. 17 ELU activation function with alpha equal to one, source: author

ELU is different from ReLU by the alpha constant that defines function smoothness at the negative part.

Advantages of ELU:

- Better performance than ReLU.
- Avoid the problem of vanishing gradient.
- Avoid the problem of exploding gradient.
- Avoid the problem of “dead ReLu”.

4.3.1.2 Layer weight initializers

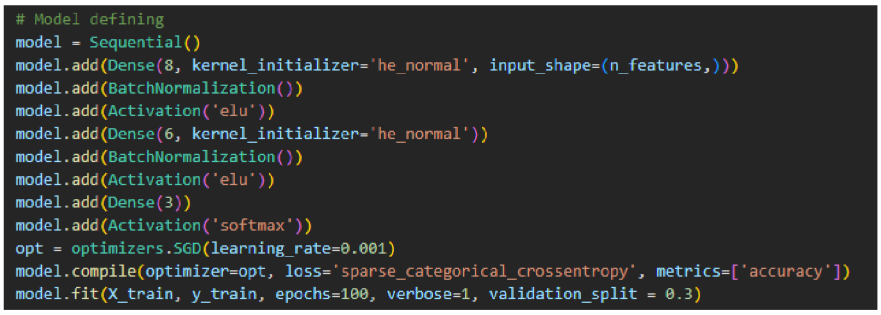
Learning of machine learning model is calculating using statistics and mathematics the weight between nodes. Layer weight initializers are a part of Keras API that helps to initiate the weight before starting learning. I will be using a He_normal initializer that uses standard deviation and a number of input nodes to calculate the weight.

4.3.1.3 Batch Normalization

In deep learning, normalizing makes starts with training over the first step of learning. When our data are going deeper into the training algorithm, the distribution of input layers is changing and a result of it could be a vanishing gradient. To avoid that problem, I will use Batch Normalization from Keras API.

4.3.1.4 Optimizer

Optimizer is the algorithm of the machine learning model that changes hyperparameters based on iteration. It is used to decrease the loss and increase the accuracy of the model. I will use the stochastic gradient descent method.



```python
# Model defining
model = Sequential()
model.add(Dense(8, kernel_initializer='he_normal', input_shape=(n_features,)))
model.add(BatchNormalization())
model.add(Activation('elu'))
model.add(Dense(6, kernel_initializer='he_normal'))
model.add(BatchNormalization())
model.add(Activation('elu'))
model.add(Dense(3))
model.add(Activation('softmax'))
opt = optimizers.SGD(learning_rate=0.001)
model.compile(optimizer=opt, loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=100, verbose=1, validation_split = 0.3)
```

Fig. 18 Defining the tunned model, source: author

5 Results and Discussion

5.1 Learning process

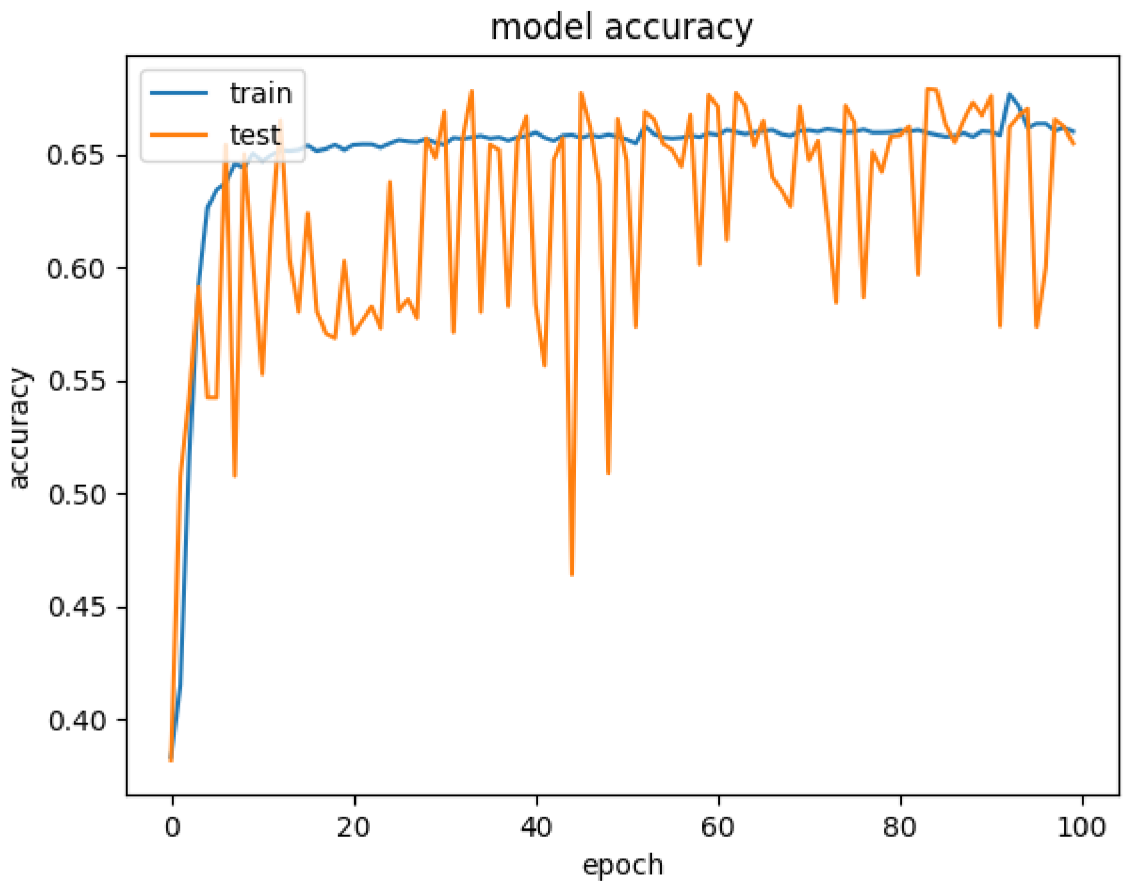


Fig. 19 Accuracy of the tunned model depends on the number of epochs, source: author

Accuracy metrics answer the question of how accurate the model is. In my case, how many samples are classified correctly? Fig. 19 represented the learning process of the model based on the accuracy of the model. Accuracy is slowly increasing with increasing epochs. The resulting accuracy is equal to 68.2%.

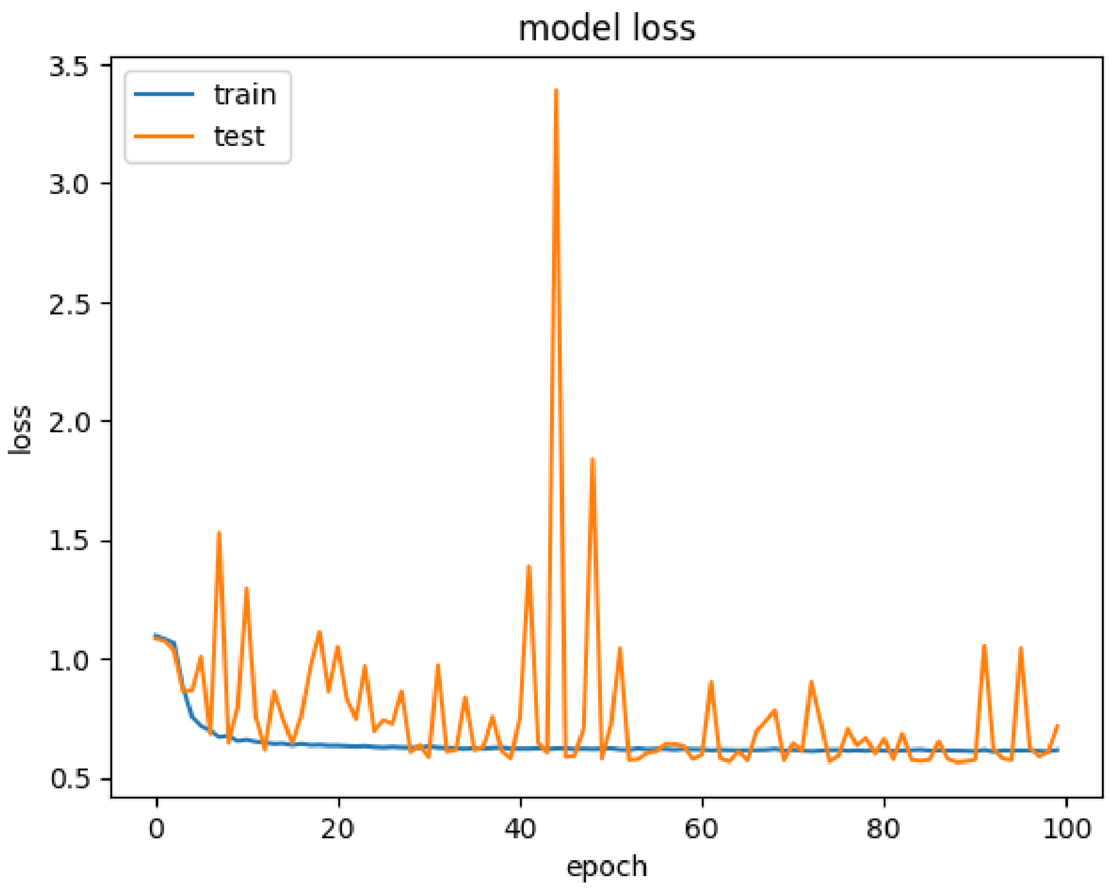


Fig. 20 Loss function of the tunned model depends on a number of epochs., source: author

The loss metric of the machine learning process shows how many samples are classified incorrectly. In an excellent learned model, the loss metric tends to be zero. Fig. 20 represented the learning process of the model based on the loss of the model. The decreasing of loss with the increasing of an epoch is good behaviour of the learning process.

5.2 Metrics

To evaluate and make a conclusion about the result of machine learning, I am going to use several metrics. Based on the type of machine learning task, metrics itself and the method of computation may vary. Also, before starting to talk about the metrics for multiclassification task, let clarify key terms related to this case of machine learning tasks.

In binary classification problems, where we are trying to predict one of two possible outcomes (e.g. whether a patient has a disease or not), we use four key terms to evaluate the accuracy of our predictions. These terms are:

- True Positive (TP): A true positive is a correct positive prediction made by the model. In other words, the model correctly identifies a positive instance as positive. For example, if the model predicts that a patient has a disease and the patient actually has the disease, then it is a true positive.
- True Negative (TN): A true negative is a correct negative prediction made by the model. In other words, the model correctly identifies a negative instance as negative. For example, if the model predicts that a patient does not have a disease and the patient actually does not have the disease, then it is a true negative.
- False Positive (FP): A false positive is an incorrect positive prediction made by the model. In other words, the model predicts that an instance is positive when it is actually negative. For example, if the model predicts that a patient has a disease, but the patient does not actually have the disease, then it is a false positive.
- False Negative (FN): A false negative is an incorrect negative prediction made by the model. In other words, the model predicts that an instance is negative when it is actually positive. For example, if the model predicts that a patient does not have a disease, but the patient actually has the disease, then it is a false negative.

These four terms are used to create a confusion matrix, which is a table that summarizes the model's predictions. The confusion matrix helps us to evaluate the accuracy of the model and to understand its strengths and weaknesses. The relationship between true positive, true negative, false positive, and false negative is important because they help us to evaluate the overall accuracy of the model and to identify areas where the model needs improvement.

5.2.1 Precision

Precision is the ratio of true positive (TP) predictions to the total number of positive predictions (both true positives and false positives). Precision measures how well the model correctly identifies positive instances out of all predicted positive instances. It is calculated as TP/(TP + FP).

To calculate precision for the multiclass classification task I will use two approaches:

Macro averaged precision: that calculates precision for each class and then gets the average.

Micro average precision: calculates wise true positives and false positives and then uses them to calculate the precision.

The macro-averaged precision of the model is 77.21%, which indicates that on average, the model is correct in 77.21% of the cases across all classes.

The micro-averaged precision of the model is 68.21%, which indicates that the model is correct in 68.21% of the cases overall, regardless of the class.

It is worth noting that the micro-averaged precision is lower than the macro-averaged precision, which suggests that the model may be better at predicting some classes than others.

5.2.2 Recall

Recall is the ratio of true positive (TP) predictions to the total number of actual positive instances (both true positives and false negatives). Recall measures how well the model correctly identifies positive instances out of all actual positive instances. It is calculated as TP/(TP + FN).

For the multiclass classification task approach to calculate similar to precision.

The macro-averaged precision of the model is 77.21%, which indicates that on average, the model is correct in 77.21% of the cases across all classes.

The micro-averaged precision of the model is 68.21%, which indicates that the model is correct in 68.21% of the cases overall, regardless of the class.

It is worth noting that the micro-averaged precision is lower than the macro-averaged precision, which suggests that the model may be better at predicting some classes than others.

5.2.3 F1 score

F1-Score computes the trade-off between precision and recall. Mathematically, it is the harmonic mean of precision and recall. [16] It is a balanced measure of precision and recall that provides a single score that summarizes the model's performance. The F1-score is calculated as 2*(precision*recall)/(precision+recall).

For the multiclass classification task approach to calculate similar to precision.

The macro-averaged precision of the model is 77.21%, which indicates that on average, the model is correct in 77.21% of the cases across all classes.

The micro-averaged precision of the model is 68.21%, which indicates that the model is correct in 68.21% of the cases overall, regardless of the class.

It is worth noting that the micro-averaged precision is lower than the macro-averaged precision, which suggests that the model may be better at predicting some classes than others.

5.2.4 Area under the ROC(Receiver Operating Characteristic) curve

AUC-ROC is a performance metric that measures the ability of a binary classification model to distinguish between positive and negative classes. The ROC curve is a plot of true positive rate (sensitivity) against false positive rate (1-

specificity) for different threshold values. The AUC-ROC is the area under the ROC curve and provides a single score that summarizes the model's performance.

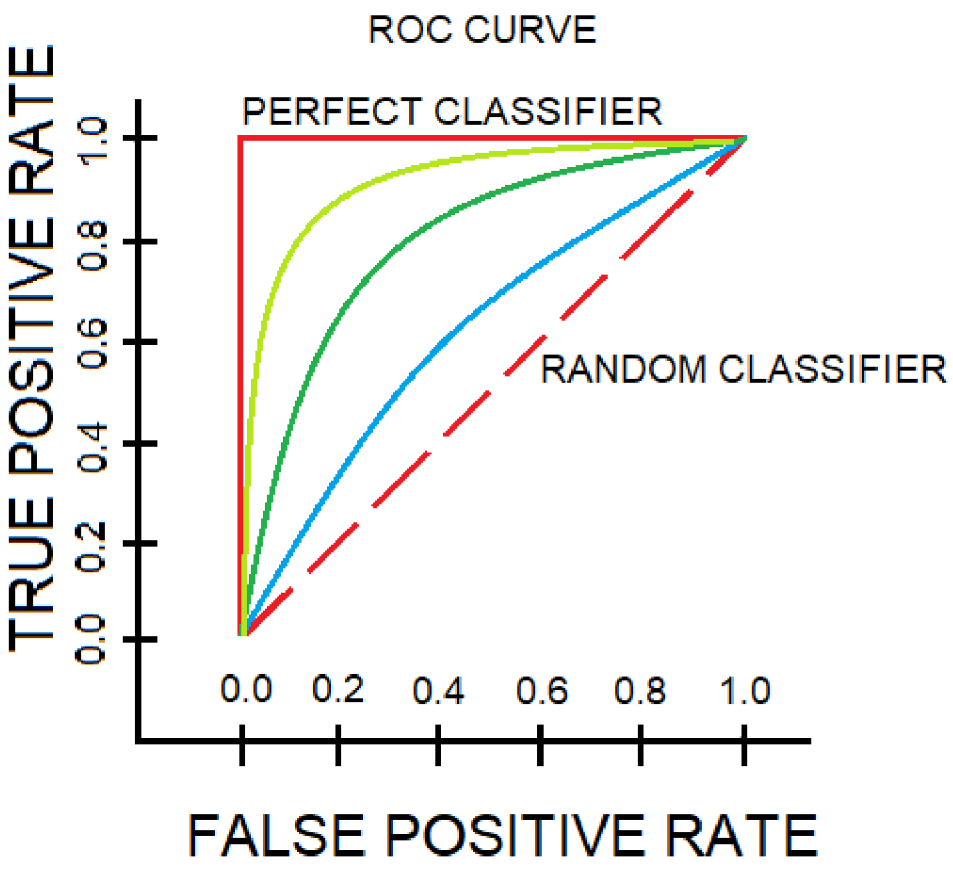


Fig. 21 Receiver Operating Characteristic curve, source: author

Tab. 1., AUC – ROC curve of the trained model

Class – 0, Forward	Class – 1, Future	Class – 2, Swap
94.02%	52.83%	75.22%

The AUC-ROC scores for the three classes also provide insights into the model's performance. Class 0 has the highest AUC-ROC score, indicating that the model is performing well on this class. However, class 1 has a lower AUC-ROC score, indicating that the model may be struggling with this class. Class 2 has a moderate AUC-ROC score, suggesting that the model may be performing moderately well on this class.

5.2.5 Confusion matrix

A confusion matrix is a table that summarizes the predictions made by a classification model. It shows the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) for each class. The confusion matrix is useful for evaluating the performance of a model and identifying areas for improvement.

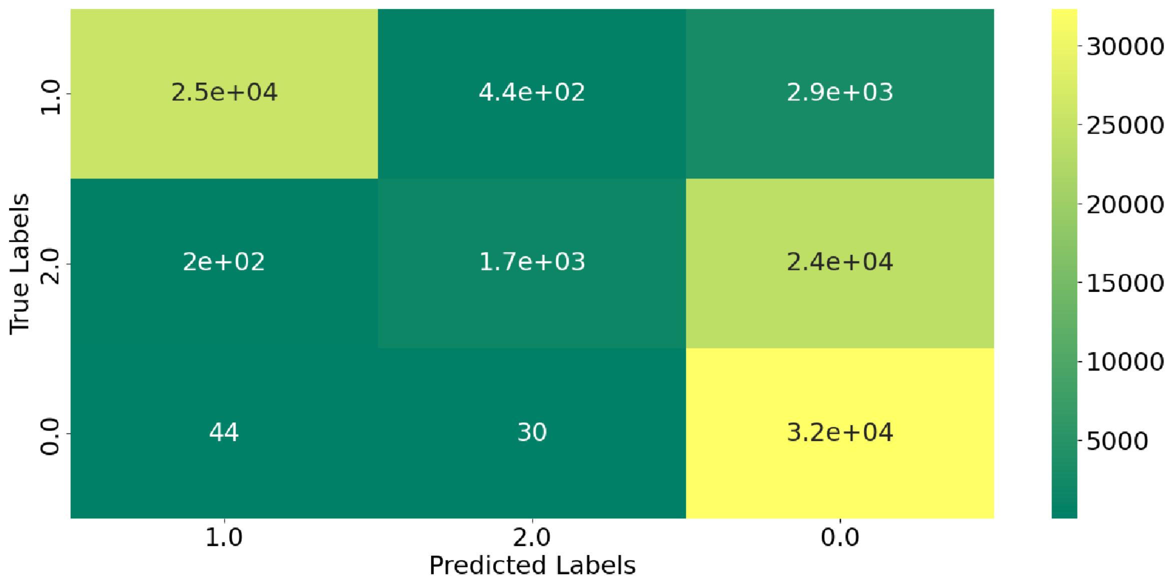


Fig. 22 Confusion matrix, source: author

The confusion matrix provides further insights into the model's performance on each class. We can see that the model is performing well on class 0 and class 2 but is struggling with class 1. Specifically, the model has a high number of false positives and false negatives for class 1, which is likely contributing to the lower micro-averaged precision and AUC-ROC score for this class.

To address these issues, possible solutions include adjusting the model parameters, using a different algorithm, collecting more data for class 1, or applying data augmentation techniques to balance the data. Additionally, we can investigate the features and data for class 1 to identify any patterns or outliers that could be affecting the model's performance. By making adjustments and improving the model's performance on class 1, we can improve the overall micro-averaged precision and AUC-ROC score of the model.

6 Conclusion

The objective of this thesis was to design and develop a machine-learning model to facilitate the ETL process. The theoretical section outlined the potential applications of the model and the theoretical basis of the deep learning multiclass classification algorithm. The practical section detailed the process of generating a dataset, conducting exploratory data analysis, and preprocessing the data for the machine learning algorithm. The initial version of the model produced an accuracy of 44.52%, which was attributed to mismatched hyperparameters. To address this, hyperparameters were fine-tuned by changing the activation function, layer weight initializer, adding the Batch Normalization layer, and changing the optimizer. The revised model achieved an accuracy of 68,2%, with decreasing loss function over epochs.

Metrics such as precision, recall, F1 score, and AUC-ROC curve were employed to evaluate the model's performance, revealing an overall good fit for the training data. While the model was found to predict swaps more successfully than forward and future transactions, a high number of discrepancies between class 0 (forward) and class 2 (swap) were noted, potentially attributed to the encoding technique used.

This thesis provides a reference for deep learning classification models, with potential for the addition of NLP algorithms for improved embedding and encoding of categorical variables. Improvements to the model may be made by implementing a learning rate schedule, callback parameter for model compilation, and the k-fold technique for better validation. In addition, for field-to-column classification, all data should be utilized as input features. However, an increased number of input features may result in a slower learning process due to the complex neural connections within the model.

The prediction time using the model was found to be only 4.5 seconds, significantly faster than manual scrolling through the dataset. This highlights the potential benefits of machine learning for automating the ETL process.

7 References

[1] Dictionary of Merriam-Webster: Data. [accepted 28.10.2022] available at: <https://www.merriam-webster.com/dictionary/data>

[2] Kennedy N., 2008: Google processes over 20 petabytes of data per day [accepted 28.10.2022] available at: <https://www.niallkennedy.com/blog/2008/01/google-mapreduce-stats.html>

[3] Vassiliadis P. and Simitsis A., 2009: Extraction, transformation, and loading. In Encyclopedia of Database Systems.

[4] Vostrovský V., Hanzlík P., 2022: DATABASE SYSTEMS (EIE36E): Database Theory.

[5] Inmon W., 2005: Building the data warehouse. John Wiley & sons

[6] Silaparasetty N., 2020: Machine Learning Concepts with Python and the Jupyter Notebook Environment: Using Tensorflow 2.0

[7] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I., 2017: Attention Is All You Need

[8] Murphy K. P., 2021: Probabilistic Machine Learning: An Introduction

[9] Müller A. and Guido S., 2016: Introduction to Machine Learning with Python

[10] Mousavi S., Schukat M., Howley E., 2018: Deep Reinforcement Learning: An Overview

[11] GÉRON A. Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. Beijing; Boston; Farnham; Sevastopol; Tokyo: O'Reilly, 2019. ISBN 978-1-492-03264-9.

[12] Loss Functions [accepted 21.01.2023] available at: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#cross-entropy

[13] A Brief History of Python [accepted 27.03.2023] available at: https://learnpython.com/blog/history-of-python/

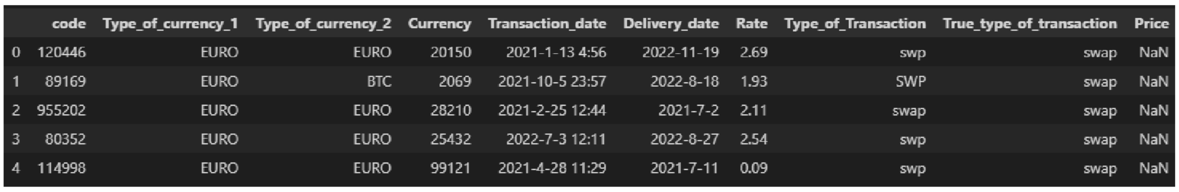
[14] Dr. Wiley J., Pace L., 2015: Beginning R. An Introduction to Statistical Programming.

[15] Anthony F., 2015: Mastering pandas.

[16] Y. Wei, J. Jang-Jaccard, F. Sabrina, A. Singh, W. Xu and S. Camtepe, "AE-MLP: A Hybrid Deep Learning Approach for DDoS Detection and Classification," in IEEE Access, vol. 9, pp. 146810-146821, 2021, doi: 10.1109/ACCESS.2021.3123791.

8 Appendix

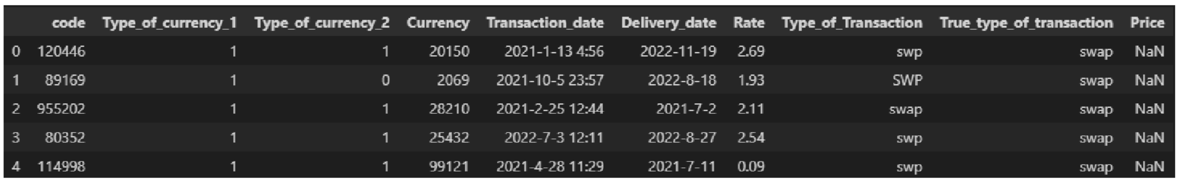
8.1 Appendix 1.



	code	Type_of_currency_1	Type_of_currency_2	Currency	Transaction_date	Delivery_date	Rate	Type_of_Transaction	True_type_of_transaction	Price
0	120446	EURO	EURO	20150	2021-1-13 4:56	2022-11-19	2.69	swp	swap	NaN
1	89169	EURO	BTC	2069	2021-10-5 23:57	2022-8-18	1.93	SWP	swap	NaN
2	955202	EURO	EURO	28210	2021-2-25 12:44	2021-7-2	2.11	swap	swap	NaN
3	80352	EURO	EURO	25432	2022-7-3 12:11	2022-8-27	2.54	swp	swap	NaN
4	114998	EURO	EURO	99121	2021-4-28 11:29	2021-7-11	0.09	swp	swap	NaN

Appendix 1 Concatenated Dataset, source: author

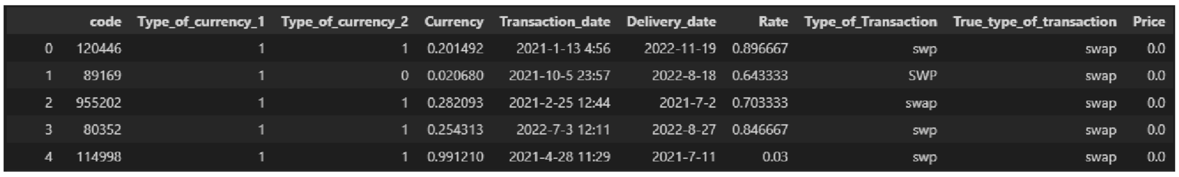
8.2 Appendix 2.



	code	Type_of_currency_1	Type_of_currency_2	Currency	Transaction_date	Delivery_date	Rate	Type_of_Transaction	True_type_of_transaction	Price
0	120446	1	1	20150	2021-1-13 4:56	2022-11-19	2.69	swp	swap	NaN
1	89169	1	0	2069	2021-10-5 23:57	2022-8-18	1.93	SWP	swap	NaN
2	955202	1	1	28210	2021-2-25 12:44	2021-7-2	2.11	swap	swap	NaN
3	80352	1	1	25432	2022-7-3 12:11	2022-8-27	2.54	swp	swap	NaN
4	114998	1	1	99121	2021-4-28 11:29	2021-7-11	0.09	swp	swap	NaN

Appendix 2 Result of label encoding of Type of currency 1 and Type of currency 2 columns, source: author

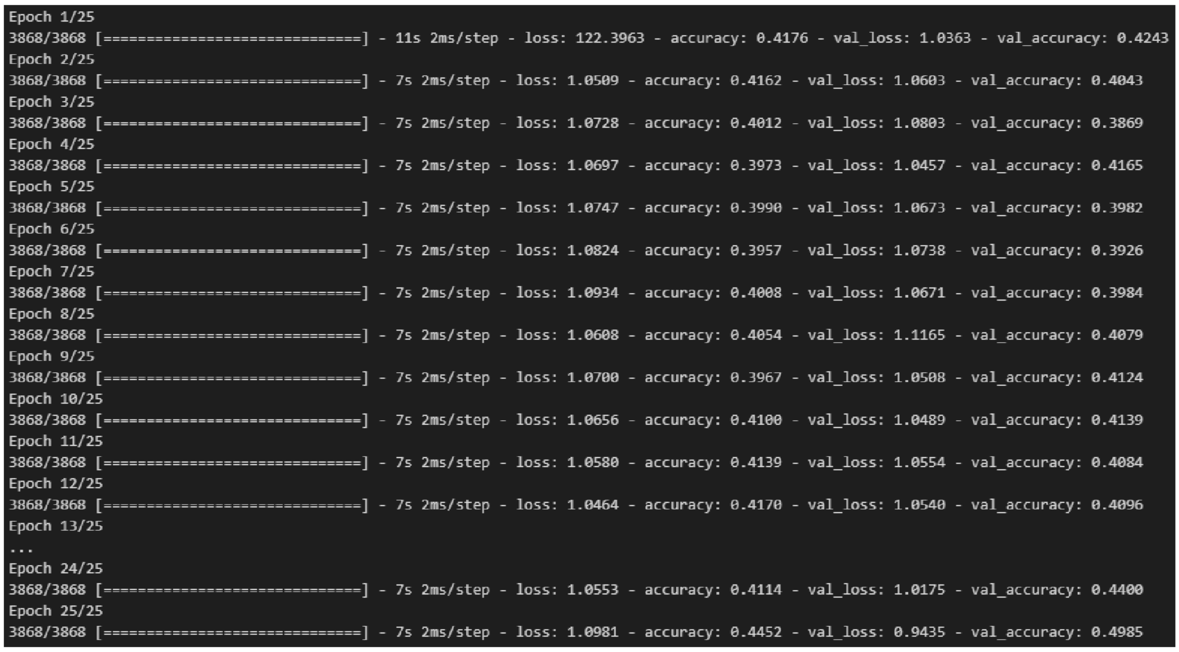
8.3 Appendix 3.



	code	Type_of_currency_1	Type_of_currency_2	Currency	Transaction_date	Delivery_date	Rate	Type_of_Transaction	True_type_of_transaction	Price
0	120446	1	1	0.201492	2021-1-13 4:56	2022-11-19	0.896667	swp	swap	0.0
1	89169	1	0	0.020680	2021-10-5 23:57	2022-8-18	0.643333	SWP	swap	0.0
2	955202	1	1	0.282093	2021-2-25 12:44	2021-7-2	0.703333	swap	swap	0.0
3	80352	1	1	0.254313	2022-7-3 12:11	2022-8-27	0.846667	swp	swap	0.0
4	114998	1	1	0.991210	2021-4-28 11:29	2021-7-11	0.03	swp	swap	0.0

Appendix 3 Result of MinMax encoding for Price, Currency and Rate columns, source: author

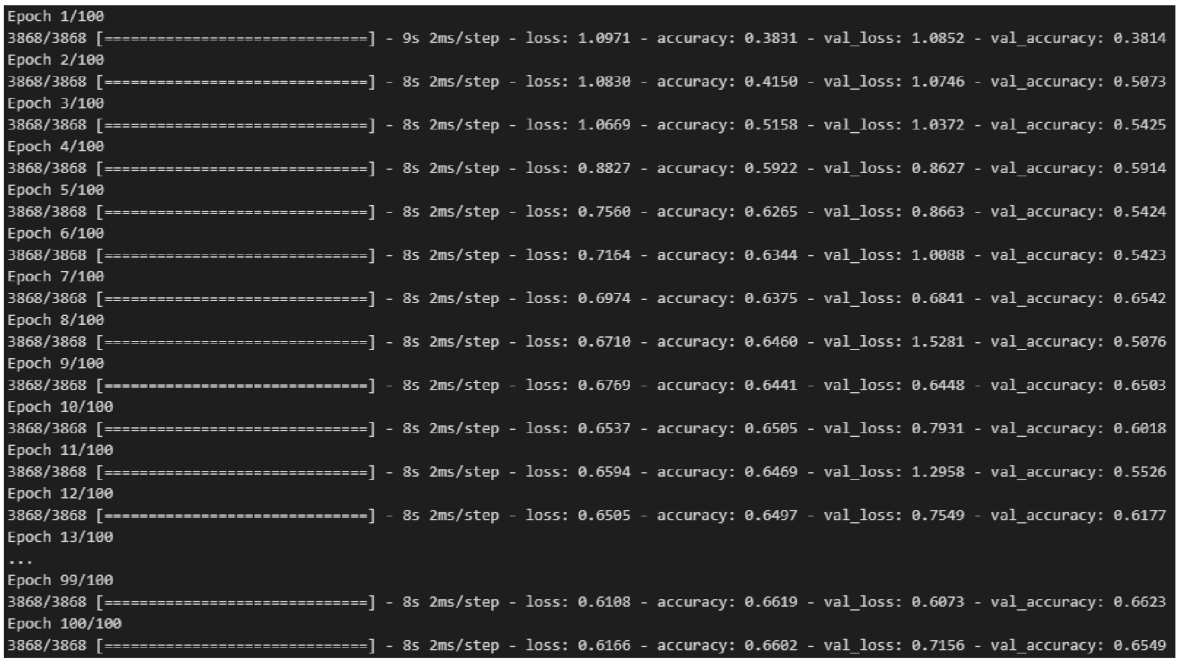
8.4 Appendix 4.



```
Epoch 1/25
3868/3868 [==============================] - 11s 2ms/step - loss: 122.3963 - accuracy: 0.4176 - val_loss: 1.0363 - val_accuracy: 0.4243
Epoch 2/25
3868/3868 [==============================] - 7s 2ms/step - loss: 1.0509 - accuracy: 0.4162 - val_loss: 1.0603 - val_accuracy: 0.4043
Epoch 3/25
3868/3868 [==============================] - 7s 2ms/step - loss: 1.0728 - accuracy: 0.4012 - val_loss: 1.0803 - val_accuracy: 0.3869
Epoch 4/25
3868/3868 [==============================] - 7s 2ms/step - loss: 1.0697 - accuracy: 0.3973 - val_loss: 1.0457 - val_accuracy: 0.4165
Epoch 5/25
3868/3868 [==============================] - 7s 2ms/step - loss: 1.0747 - accuracy: 0.3990 - val_loss: 1.0673 - val_accuracy: 0.3982
Epoch 6/25
3868/3868 [==============================] - 7s 2ms/step - loss: 1.0824 - accuracy: 0.3957 - val_loss: 1.0738 - val_accuracy: 0.3926
Epoch 7/25
3868/3868 [==============================] - 7s 2ms/step - loss: 1.0934 - accuracy: 0.4008 - val_loss: 1.0671 - val_accuracy: 0.3984
Epoch 8/25
3868/3868 [==============================] - 7s 2ms/step - loss: 1.0608 - accuracy: 0.4054 - val_loss: 1.1165 - val_accuracy: 0.4079
Epoch 9/25
3868/3868 [==============================] - 7s 2ms/step - loss: 1.0700 - accuracy: 0.3967 - val_loss: 1.0508 - val_accuracy: 0.4124
Epoch 10/25
3868/3868 [==============================] - 7s 2ms/step - loss: 1.0656 - accuracy: 0.4100 - val_loss: 1.0489 - val_accuracy: 0.4139
Epoch 11/25
3868/3868 [==============================] - 7s 2ms/step - loss: 1.0580 - accuracy: 0.4139 - val_loss: 1.0554 - val_accuracy: 0.4084
Epoch 12/25
3868/3868 [==============================] - 7s 2ms/step - loss: 1.0464 - accuracy: 0.4170 - val_loss: 1.0540 - val_accuracy: 0.4096
Epoch 13/25
...
Epoch 24/25
3868/3868 [==============================] - 7s 2ms/step - loss: 1.0553 - accuracy: 0.4114 - val_loss: 1.0175 - val_accuracy: 0.4400
Epoch 25/25
3868/3868 [==============================] - 7s 2ms/step - loss: 1.0981 - accuracy: 0.4452 - val_loss: 0.9435 - val_accuracy: 0.4985
```

Appendix 4 The learning process of the initial model, source: author

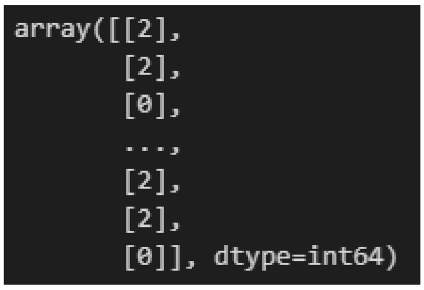
8.5 Appendix 5.



```
Epoch 1/100
3868/3868 [==============================] - 9s 2ms/step - loss: 1.0971 - accuracy: 0.3831 - val_loss: 1.0852 - val_accuracy: 0.3814
Epoch 2/100
3868/3868 [==============================] - 8s 2ms/step - loss: 1.0830 - accuracy: 0.4150 - val_loss: 1.0746 - val_accuracy: 0.5073
Epoch 3/100
3868/3868 [==============================] - 8s 2ms/step - loss: 1.0669 - accuracy: 0.5158 - val_loss: 1.0372 - val_accuracy: 0.5425
Epoch 4/100
3868/3868 [==============================] - 8s 2ms/step - loss: 0.8827 - accuracy: 0.5922 - val_loss: 0.8627 - val_accuracy: 0.5914
Epoch 5/100
3868/3868 [==============================] - 8s 2ms/step - loss: 0.7560 - accuracy: 0.6265 - val_loss: 0.8663 - val_accuracy: 0.5424
Epoch 6/100
3868/3868 [==============================] - 8s 2ms/step - loss: 0.7164 - accuracy: 0.6344 - val_loss: 1.0088 - val_accuracy: 0.5423
Epoch 7/100
3868/3868 [==============================] - 8s 2ms/step - loss: 0.6974 - accuracy: 0.6375 - val_loss: 0.6841 - val_accuracy: 0.6542
Epoch 8/100
3868/3868 [==============================] - 8s 2ms/step - loss: 0.6710 - accuracy: 0.6460 - val_loss: 1.5281 - val_accuracy: 0.5076
Epoch 9/100
3868/3868 [==============================] - 8s 2ms/step - loss: 0.6769 - accuracy: 0.6441 - val_loss: 0.6448 - val_accuracy: 0.6503
Epoch 10/100
3868/3868 [==============================] - 8s 2ms/step - loss: 0.6537 - accuracy: 0.6505 - val_loss: 0.7931 - val_accuracy: 0.6018
Epoch 11/100
3868/3868 [==============================] - 8s 2ms/step - loss: 0.6594 - accuracy: 0.6469 - val_loss: 1.2958 - val_accuracy: 0.5526
Epoch 12/100
3868/3868 [==============================] - 8s 2ms/step - loss: 0.6505 - accuracy: 0.6497 - val_loss: 0.7549 - val_accuracy: 0.6177
Epoch 13/100
...
Epoch 99/100
3868/3868 [==============================] - 8s 2ms/step - loss: 0.6108 - accuracy: 0.6619 - val_loss: 0.6073 - val_accuracy: 0.6623
Epoch 100/100
3868/3868 [==============================] - 8s 2ms/step - loss: 0.6166 - accuracy: 0.6602 - val_loss: 0.7156 - val_accuracy: 0.6549
```

Appendix 5 The learning process of the tunned model, source: author

8.6 Appendix 6



```
array([[2],
       [2],
       [0],
       ...,
       [2],
       [2],
       [0]], dtype=int64)
```

Appendix 6 Predicted values on the test dataset, source: author