Detecting Fraudulent Bank Accounts: A Data Mining and Machine Learning Approach

by

Raquel Beltrán Barba

A thesis submitted in conformity with the requirements
for the MSc in Economics, Finance and Computer Science

Universidad de Huelva & Universidad Internacional de Andalucía

Huelva, September 2023

Detecting Fraudulent Bank Accounts: A Data Mining and Machine Learning Approach

Raquel Beltrán Barba
MSc in Economics, Finance and Computer Science

Supervised by:
Prof. Dr. Antonio J. Tallón Ballesteros
Universidad de Huelva

Abstract

Nowadays, the detection of fraudulent activities within the banking sector has become a critical concern for financial institutions. The rise of electronic transactions and the escalating complexity of techniques employed by financial criminals show the great importance of early and precise fraudulent bank accounts identification. This is essential for maintaining the integrity of the financial system.

Therefore, this Master's Thesis will involve working in a dataset designed to simulate the kind of information typically possessed by a banking institution. The main intention is to check out the dataset in search for characteristics that make a transaction or an account holder suspicious of fraud. To accomplish this, Data Mining and Machine Learning techniques will be used for an exhaustive examination of the dataset, as well as applying advanced classification algorithms. These algorithms will play a pivotal role in effectively distinguishing between transactions that ultimately prove to be fraudulent and those that are legitimate, posing no risk to the financial institution.

Success in this attempt will grant benefits not only for the specific banking institution in question but will also contribute significantly to fortifying the financial system. It will enhance the ability to detect and prevent fraudulent activities in an environment that is becoming progressively more digital and complex.

JEL Classification C38, C45, E51, G21
Keywords Machine Learning, Supervised Learning, SMOTE, Logistic Regression, Decision Tree, Neural Networks

Resumen

En la era digital en la que nos encontramos, la detección de actividades fraudulentas en el ámbito bancario se ha convertido en una preocupación de suma importancia para las instituciones financieras. Esto se debe a la amplia expansión de las transacciones electrónicas y a la creciente sofisticación de los métodos utilizados por los delincuentes financieros. La capacidad de identificar de manera temprana y precisa cuentas bancarias fraudulentas se ha vuelto esencial para preservar la integridad y la confianza en el sistema financiero.

Por lo tanto, en este Trabajo de Fin de Máster, se abordará el análisis de un conjunto de datos que simula el tipo de información que podría poseer una entidad bancaria. El objetivo principal es identificar las características que hacen que una transacción o el titular de una cuenta se vuelvan sospechosos de estar involucrados en actividades fraudulentas. Para llevar a cabo esta tarea, se aplicarán técnicas de Minería de Datos y Aprendizaje Automático, las cuales permitirán explorar exhaustivamente el conjunto de datos y utilizar algoritmos de clasificación avanzados. Estos algoritmos serán cruciales para diferenciar de manera efectiva las transacciones que resultan ser fraudulentas de aquellas que son legítimas y no representan un riesgo para la institución financiera.

El éxito en este ejercicio no solo beneficiará a la entidad bancaria en cuestión, sino que también contribuirá al fortalecimiento del sistema financiero en su conjunto, al aumentar la capacidad de detección y prevención de actividades fraudulentas en un entorno cada vez más digital y complejo.

Acknowledgements

To my loving parents, for the countless opportunities and unwavering support they have always provided me.

To my sister and friends, who have always had faith in me and I know will always be there.

To my family, who always reminds me of my worth and celebrates my successes with immense pride.

And a special thanks to my supervisor, Antonio Tallón, for his trust and dedication from day one, for helping me bring this project to fruition.

Table of Contents

List of Tables

List of Figures

1 Introduction

In the last few decades, the most significant changes that humans have experienced have been thanks to (or caused by) technology and its advancements. These advancements allow us to approach problems from a different perspective that could not be addressed before, improve everyday life situations, or simplify the search for solutions to some problems. Precisely, in recent years, the technology that has brought about a major change in the present day is Artificial Intelligence (AI) and Machine Learning as part of it.

Machine Learning is the study of statistical models and algorithms used to enhance the performance of certain programs by learning through experience. These algorithms are created from data, but they go beyond that by learning which mistakes not to repeat and what successes to learn from [27] [18].

This intelligence needed to solve problems inherent in datasets is constantly evolving, and over time it has become evident that an increasing number of real-world problems can be addressed through AI.

These algorithms aim to learn the relationships within data that humans may not necessarily perceive. This is a significant advantage, as these analytical models used to require prior knowledge of these relationships and procedures before working with the data. By now having systems that can freely search for these aspects, we save time and reduce errors in their predictions and studies[2].

Depending on the type of problem to be solved and the dataset itself, different learning techniques are employed [21] [3]:

- Supervised Learning: The algorithm generates a function that maps inputs to desired outputs (labeled dataset), which helps the algorithm to learn and establish relationships between inputs and outputs. It works for both regression and classification tasks.
- Unsupervised Learning: Unsupervised Learning is used when the dataset lacks corresponding output labels. Algorithms in this category seek to model and analyze the inherent characteristics of instances to gain a deeper understanding.
- Reinforcement Learning: The algorithm is rewarded or penalized while it is actively working. The primary goal is to maximize the accumulative reward over time.
- Semi-Supervised Learning: Semi-Supervised Learning combines the advantages of both Supervised and Unsupervised Learning. In this approach, a portion of the dataset has known output labels, while a larger portion does not.

There are many other techniques that can be applied to algorithms, such as *Transfer Learning, Self-taught Learning, Active Learning* or *Targeted Learning*, but the ones described above are the most popular and used.

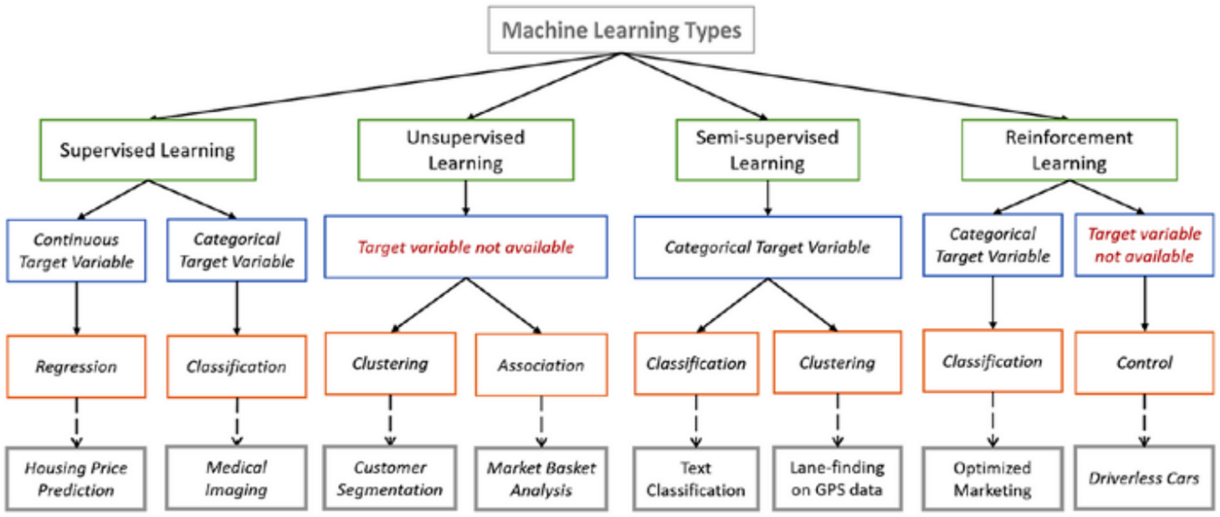

Figure 1: Machine Learning Techniques

1.1 Context

Detecting fraudulent requests and transactions in the banking world represents a problem of great importance. As technology advances and is increasingly used to our advantage, there are those who manage to find new ways to fool banking systems and their users. Fraud does not only result in financial losses for the banking institution and its clients but also destroys the trust in the financial sector as a whole.

Fortunately, advancements in artificial intelligence and machine learning empower us to analyze vast amounts of data, enabling the detection of patterns in these fraudulent transactions (or transaction requests).

1.2 Goals

The goals of this Master's Thesis are to achieve the proper preparation and treatment of the data to identify the most effective Machine Learning technique for addressing and describing the issue of bank fraud at hand.

Additionally, we will look for a predictive model that can accurately determine the value of the response attribute by using the predictor attributes considered within the dataset to achieve the highest possible level of accuracy.

2 Description of the Data

The dataset under study was published in **NeurlPS 2022** and consists of a collection of databases that simulate realistic information a bank might have about its bank accounts and transactions [16].

The offered datasets include:

- Base: This dataset aims to represent the original database of a bank as accurately as possible. This is the dataset that will be studied in this Master's Thesis.
- Variant I: Similar to the base dataset but exhibits a higher group size disparity than base dataset.
- Variant II: Unlike the base dataset, this one has a higher prevalence of disparity.
- Variant III: This dataset features greater separability for one of the groups.
- Variant IV: In this dataset, the specified training set has a higher prevalence of disparity.
- Variant V: Similar to the previous one, the specified training set has greater class separability.

The dataset comprises one million instances with 32 attributes and two classes.

The information acquired for each instance is:

- *fraud_bool*: Binary attribute. It is marked as 1 if the bank account is fraudulent, and 0 otherwise.
- *income*: Annual salary of the person who owns the bank account, measured in quantiles.
- *name_email_similarity*: Similarity between the name of the account holder and their email.
- *prev_address_months_count*: Number of months the email associated with the account was different from the current one.
- *current_address_months_count*: Number of months the current email address has been registered.
- *customer_age*: Account owner's age, measured in decades.
- *days_since_request*: Number of days since the last request was made.
- *intended_balcon_account*: Initial amount of money transferred.
- *payment_type*: Payment type, with five possible options (AA, EB, AC, AD, AE).
- *zip_count_4w*: Number of requests made in the last 4 weeks with the same postal code.
- *velocity_6h*: Average number of requests per hour in the last 6 hours.
- *velocity_24h*: Average number of requests per hour in the last 24 hours.

- *velocity_4w*: Average number of requests per hour in the last 4 weeks.
- *bank_branch_count_8w*: Total number of requests at the bank branch where the instance's request is made in the last 8 weeks.
- *date_of_birth_distinct_emails_4w*: Number of email addresses with the same date of birth that have requested in the last 4 weeks.
- *employment_status*: Employment status of the applicant, with 7 possible options (CA, CB, CC, CD, CE, CF, CG).
- *credit_risk_score*: Internal score of application risk.
- *email_is_free*: Binary attribute. It is marked as 1 if the email domain is public, and 0 otherwise.
- *housing_status*: Residential status of the applicant, with 7 possible options (BA, BB, BC, BD, BE, BF, BG).
- *phone_home_valid*: Binary attribute. It indicates 1 if the associated home phone number is valid, and 0 otherwise.
- *phone_mobile_valid*: Binary attribute. It indicates 1 if the associated mobile phone number is valid, and 0 otherwise.
- *bank_months_count*: Number of months of the applicant's previous account (if applicable).
- *has_other_cards*: Binary attribute. It is marked as 1 if the applicant has more cards from the same bank, and 0 otherwise.
- *proposed_credit_limit*: Credit limit proposed by the applicant.
- *foreign_request*: Binary attribute. It is set to 1 if the request is of foreign origin, and 0 otherwise.
- *source*: Online source of the request, taking two possible values: INTERNET or APP.
- *session_length_in_minutes*: Length of user session in banking website in minutes.
- *device_os*: Operative system of device that made request. Possible values are: Windows, Macintosh, Linux, X11, or other.
- *keep_alive_session*: Binary attribute. It indicates 1 if the user accepted or declined (0) to keep the session open on the website when asked.
- *device_distinct_emails_8w* : Number of distinct emails in banking website from the used device in last 8 weeks.
- *device_fraud_count* Number of fraudulent applications with used device.
- *month*: Month in which the request was made.

2.1 Descriptive study

Out of the 32 attributes in this dataset, 19 of them are numerical, and 13 are categorical.

	Numerical Attributes		Categorical Attributes
1	income	0	fraud_bool
2	name_email_similarity	8	payment_type
3	prev_address_months_count	15	employment_status
4	current_address_months_count	17	email_is_free
5	customer_age	18	housing_status
6	days_since_request	19	phone_home_valid
7	intended_balcon_amount	20	phone_mobile_valid
9	zip_count_4w	22	has_other_cards
10	velocity_6h	24	foreign_request
11	velocity_24h	25	source
12	velocity_4w	27	device_os
13	bank_branch_count_8w	28	keep_alive_session
14	date_of_birth_distinct_emails_4w	31	month
16	credit_risk_score		
21	bank_months_count		
23	proposed_credit_limit		
26	session_length_in_minutes		
29	device_distinct_emails_8w		
30	device_fraud_count		

Table 1: Data types of the attributes

A distinctive feature of this database, and one that will need to be addressed later, is the unmistakable class imbalance, as illustrated in Figure 2. Class 0 (non-fraudulent transactions) comprises 988971 instances out of 1000000, while for class 1, representing fraudulent instances, there are only 11029.

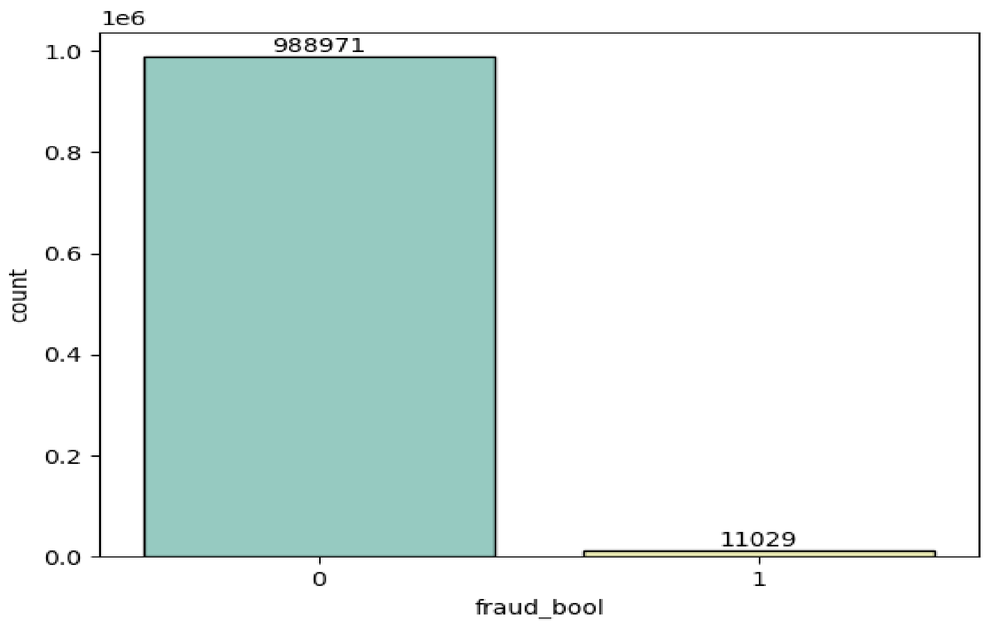

Figure 2: Instances distribution for both classes. *Self-generated image.*

This lack of representation in class 1 poses a problem when working with this dataset. Ideally, the classes should have similar distributions. To address this, the oversampling technique known as SMOTE *(Synthetic Minority Over-sampling TEchnique)* will be applied.

Before applying data treatment techniques, two attributes will be removed from the dataset as they do not provide necessary information for this study:

- *device_fraud_count*: This attribute is being removed because it only takes the value 0, indicating that none of the instances in the dataset were made with a device that had previously performed any fraudulent actions.
- *month*: This attribute was included in the dataset for studying time series, which is not considered in this Master's Thesis.

Once these two attributes are removed, the heatmap represented in Figure 3 shows us the following:

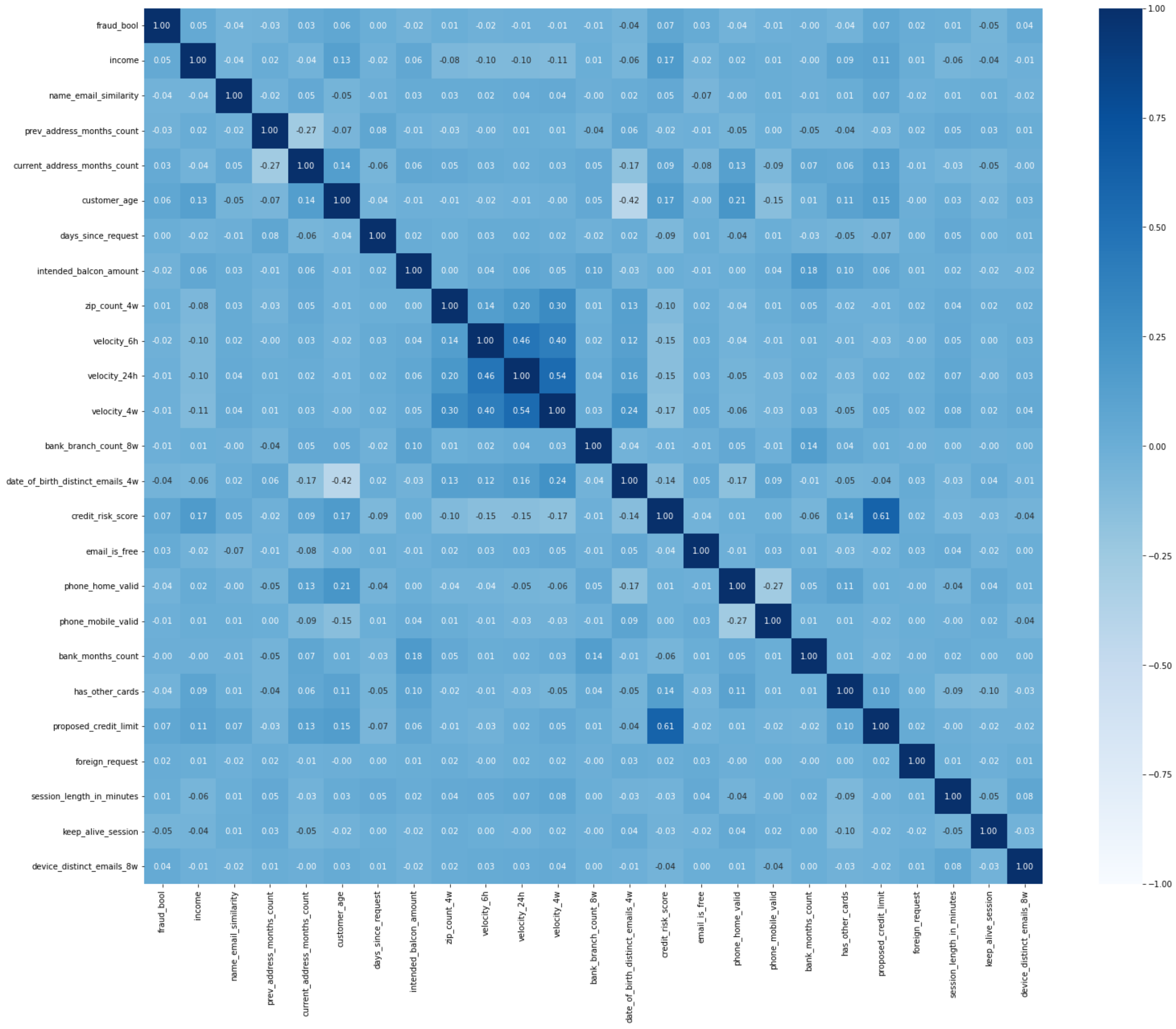

Figure 3: Heatmap for the attributes. *Self-generated image*

There are no attributes with concerning correlation, although it's worth noting the apparent (and logical) correlation among the attributes *zip_count_4w, velocity_6h, velocity_24h* and *velocity_4w*, as they all refer to similar measurements.
Another pair of attributes that exhibit a significant correlation are *credit_risk_score* and *proposed_credit_limit*, with a correlation value of 0.61.

3 Theoretical Framework

3.1 Preprocessing Techniques

3.1.1 Class rebalancing

Most classification algorithms assume that classes are balanced when, in real life (and in many existing datasets), this is not the case. To ensure that our algorithm understands the characteristics that differentiate each class as accurately as possible, it needs an adequate number of examples for each class. This is why **class rebalancing** is useful.

There are several methods to improve the performance of a classifier when facing imbalanced data, and in this Master's Thesis this problem will be addressed using the oversampling technique known as SMOTE *(Synthetic Minority Over-sampling TEchnique)*:
SMOTE works by taking instances that are close in feature space and creating a new instance that is also close within the same class. To clarify, the process works as follows: An instance from the minority class is chosen, and then k nearest neighbors (belonging to the same class) are identified within this feature space. The new instance is created by combining the initial instance with one of the k nearest neighbors that are selected [13].
One drawback of this technique is that it creates new instances without considering the majority class, which could potentially introduce ambiguous instances into the dataset[7].

To implement this technique in *Python* programming language, we will use the *imbalanced-learn* library. Particularly, we will use the function *imblearn.over_sampling.**SMOTENC***, designed for addressing imbalanced data when predictor attributes consist of both numerical and categorical features. This is in contrast to the original *imblearn.over_sampling.**SMOTE*** function, which does not handle mixed data types.

This function includes the following parameters [17]:

- *categorical_features*: List of the names of the attributes (or their positions in the DataFrame) that are categorical.
- *categorical_encoder*: By default, this parameter is deactivated, but if enabled, it applies *OneHotEncoder* to the categorical attributes. In the case of this Master's Thesis, this procedure has been applied beforehand.
- *random_state*: Seed used by the random number generator.
- *k_neighbors*: Number of neighbors to use. Its value by default is 5.

3.1.2 Data Reduction

The algorithms employed in this Master's Thesis are capable of extracting information from very large datasets, whether due to their numerous attributes or instances to be studied. However, datasets with a significant number of instances, as is the case here (1 million), increase the complexity and execution time of the study.

To address this challenge, this Master's Thesis has selected 30% of the total instances as a solution.

3.2 Machine Learning Algorithms

Machine Learning employs various algorithms to solve problems with datasets. It is a well-known fact that there is no one-size-fits-all algorithm that is the best for solving all problems. Choosing the correct algorithm depends on the specific problem at hand, the attributes and their types, and the type of model desired to describe it [18].

In this study, we will compare Machine Learning algorithms based on Supervised Learning. Supervised Learning algorithms require external guidance, comparing the algorithm's output for an instance to the actual output, in order for the algorithm to continue learning. The dataset will be split into training and test sets [18]. Supervision within the training set is crucial for training the algorithm to make accurate predictions by evaluating its output against expected results. Then, the trained algorithm is applied to the test set, where we can extract evaluation metrics [6].

3.2.1 Logistic Regression Algorithm

Logistic Regression models are statistical algorithms used to study the relationship between a binary dependent attribute (taking on the values 0 and 1) and independent attributes.

There are two main reasons for choosing a logistic regression model: the first is the ease with which parameter estimates can be interpreted as odds ratios, and the second pertains to the calculation of predicted values as probabilities of the outcome being 1 in the model's predictions.

- **Odd ratio**

  Odds ratio are defined as the measure of association between an independent attribute and the dependent attribute, explaining the influence of the relationship between them. This is often estimated by the ratio of the number of times that the event of interest occurs to the number of times that it does not [5]. This value ranges from 0 to infinite. If its value is 1, it means that there is no relationship between the attributes *(i.e., one attribute's category does not affect the other)*. If the odds ratio is less than 1, the relationship is negative, and if it is greater than 1, it is positive.

  In Logistic Regression, odds ratio are employed to assess how independent attributes influence the dependent attribute. These ratios serve as the parameters used to analyze attributes within this algorithm.

Consider a table that represents, with columns, the events of an independent attribute, and with rows, the events of the response attribute categorized accordingly [12]:

		Independient Attribute		
		0	**1**	**Total**
Dependent Attribute	**0**	a	b	a+b
	1	c	d	c+d
	Total	a+c	b+d	a+b+c+d

Table 2: Contingency table for odds ratio.

Odds ratio can be calculated and interpreted as follows:
Considering the probability of the dependent attribute taking the value 0 when the independent attribute is also 0: $\frac{\frac{a}{a+c}}{\frac{c}{a+c}} = \frac{a}{c}$.
And if we understand the probability of the dependent attribute taking the value 0 when the independent attribute is 1 as: $\frac{\frac{b}{b+d}}{\frac{c}{b+d}} = \frac{b}{d}$.

Therefore, the value of the odds ratio concerning the studied independent attribute would be:

$$OR = \frac{\frac{a}{c}}{\frac{b}{d}} = \frac{a * d}{b * c} \tag{1}$$

It would be interpreted as follows:
If the odds ratio is less than 1, individuals with a value of 0 in the studied independent attribute are less likely to have the dependent attribute take on the value 0. But when we invert this value (OR^{-1}), it implies that instances with a value of 1 in their independent attribute are $\frac{bc}{ad}$ times more likely to have the dependent attribute take on the value 0. [25]

An alternative approach to compute this odds ratio, which will be employed in this Master's Thesis, involves determining the exponential of the coefficients (also known as logits) provided by the model's regression in Python (in this case). Further elaboration on this will be provided later in the document.

3.2.2 Decision Tree-based Algorithm

Decision trees represent graphs where nodes nodes serve as filters that an instance may or may not meet. Depending on whether it satisfies these conditions or not, the instance will be classified in one way or another as it progresses towards the tree's terminal nodes [18].

Decision trees are one of the most powerful algorithms and are used in various fields such as machine learning, image processing, and pattern recognition [8]. Due to their ease of interpretation and their versatility in addressing a wide array of distinct problems, decision trees are extensively employed across various fields today [20].

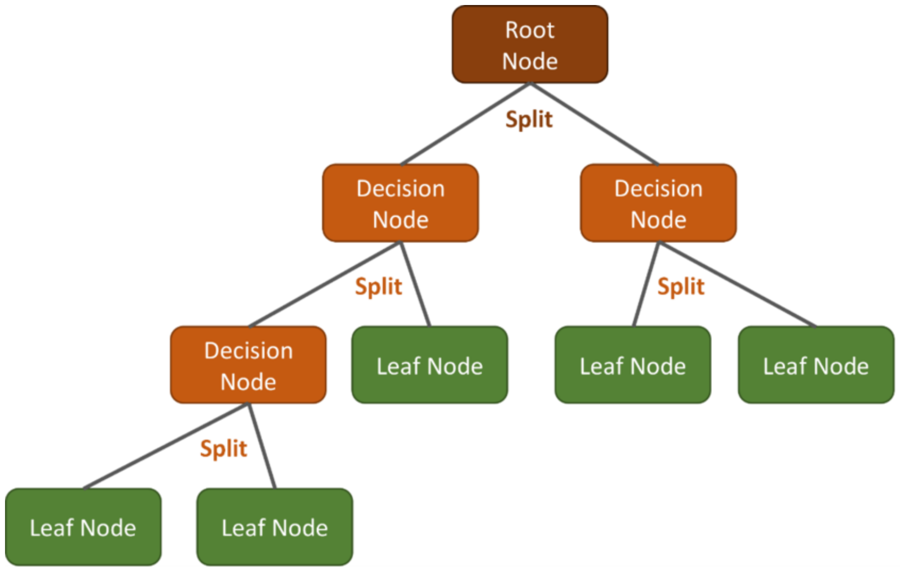

Figure 4: General Structure of a Decision Tree

There are different types of decision tree algorithms, including CART (Classification and Regression Tree), CHAID (Chi-squared Automatic Interaction Detector), QUEST (Quick, Unbiased, and Efficient Statistical Tree), CTREE (Conditional Inference Trees), ID3 (Iterative Dichotomizer Version 3), and its successor, C4.5 [8] [24].

These algorithms, primarily based on the Hunt algorithm [23], rely on primarily two steps: The first step states that if, within a node, all instances belong to the same class, that node should not be split and becomes a terminal node. The second step indicates that if there are instances with different classes, one of the variables should be selected to split the remaining instances in the node into subsets that more accurately differentiate between the classes.

The selection of the mentioned variable can be done using the Gini Index or Entropy [22]:

- **Gini Index** [24]

  The Gini Index measures the purity of a node. It is the probability of selecting two instances from a node and having them belong to different classes. Therefore, a higher Gini Index means lower purity. Hence, the attribute with the lowest Gini Index should be chosen. The Gini Index is defined as follows:

$$GINI = 1 - \sum_{i=1}^{n} (P_i)^2 \tag{2}$$

  where P_i represents the probability that an instance belongs to Class i and n is the number of instances.

- **Entropy**
  Entropy measures the level of disorder in a system, and its formula is as follows:

$$H = -\sum_{i=1}^{n} P_i * log_2(P_i) \tag{3}$$

  This way, if a node is pure (all its instances belong to the same class), the probability of belonging to Class i (P_i) equals 1, resulting in an entropy value of 0.
  When choosing a variable, both its Gini Index and Entropy value should ideally be 0.

3.2.3 Neural Network Algorithm

Deep Learning is a subset of Machine Learning that relies on Artificial Neural Networks. They comprise multiple layers, each containing a different number of neurons, tailored to the complexity and appropriate structure for the specific problem.

Artificial Neural Networks (ANNs) aim to replicate the learning process observed in the human brain, where knowledge is acquired through experience and the extraction of generic patterns from data. These networks mimic the neural structure of the human brain.

The structure of an ANN is as follows[11]:

- Input layer: This layer receives data directly from external sources.
- Hidden Layer(s): These layers do not have direct connections to the external environment and may include multiple levels.
- Output Layer: The output layer returns the desired information after making predictions. In regression tasks, it usually contains a single neuron, while in classification tasks, there will be as many neurons as there are distinct classes.

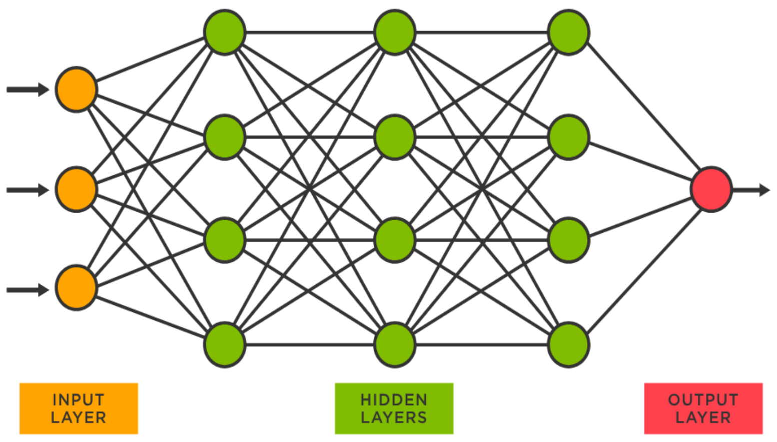

Figure 5: General Structure of a Neural Network

Each of these layers consists of units that transform the input data into information that the next layer needs to perform its assigned task. This structured approach allows the machine to learn during the data processing phase [4].

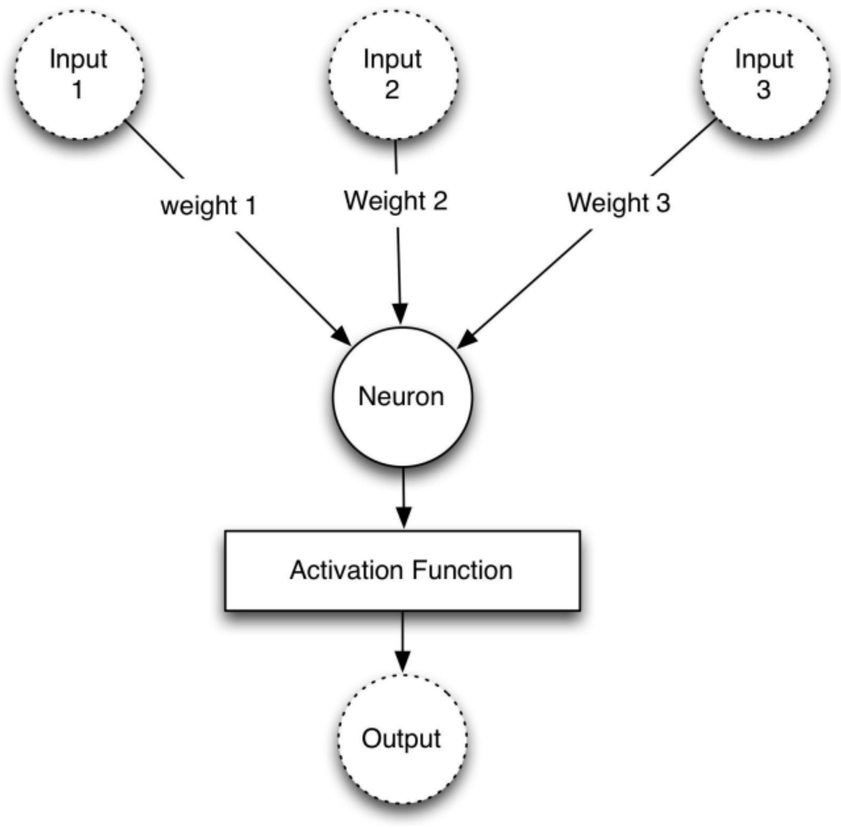

Figure 6: A closer look inside the Neural Network. [11]

Unlike conventional algorithms, ANN exhibit higher efficiency in self-training scenarios where the relationships between attributes are non-linear or subject to change over time.
The ability of ANNs to explore all these relationships makes it easier for users to quickly create models that could have been previously complex or even incomprehensible using other methods [15].

This is also a disadvantage compared to traditional algorithms because the process that a Neural Network performs may be less explainable. It will be more difficult to understand which variables are the most important or the exact steps it follows. [1]

3.3 Evaluation Metrics

Evaluation metrics will be used to determine which of the models being worked on is more accurate in its prediction [9]. Depending on the type of problem, it will be more convenient to use one set of metrics over the other. These metrics are applied to the results obtained when applying the pre-trained algorithm to the test dataset.
Before explaining the metrics used in this Master's Thesis, it is important to first understand what a Confusion Matrix is:

- **Confusion matrix**

The confusion matrix is a *kxk* table that compares the predicted classes in its rows and the actual classes in its columns. In this Master's Thesis, *k=2*. It is represented as follows:

Confusion Matrix		*Real values*	
		Positive Class (1)	*Negative Class (0)*
Predicted values	*Positive Class (1)*	True positive (TP)	False positive (FP)
	Negative Class (0)	False negative (FN)	True negative (TN)

Table 3: Confusion Matrix for Binary Classification

From Table 3, which compares the obtained results with the actual data in the dataset, we obtain the following metrics [14]:

- *Accuracy*:
  This metric computes the percentage of instances that are accurately classified. However, it has a drawback when dealing with datasets having imbalanced classes, as it might not adequately account for the minority class, as one would anticipate.
  Its formula is:
$$\frac{TP+TN}{TP+FP+TN+FN} \tag{4}$$
- *Sensitivity*:
  It represents the fraction of actual positive values that have been correctly classified. Its formula is as follows:
$$\frac{TP}{TP+FP} \tag{5}$$
- *Specificity*:
  This metric represents the fraction of actual negative values that have been correctly classified:
$$\frac{TN}{TN+FN} \tag{6}$$
- *Precision*:
  This validation metric is used to calculate the true positive predictions for the positive class. Its formula is as follows:
$$\frac{TP}{TP+FP} \tag{7}$$

- *Recall*:
  The recall metric is used to indicate the fraction of true positives among all the observations predicted as positive. Its mathematical expression is:

$$\frac{TP}{TP + FN} \tag{8}$$

- *F1-score*:
  This measure represents the consensus between precision and recall. A higher F1-score indicates better prediction performance. Its formula is as follows:

$$2 * \frac{P * R}{P + R} \tag{9}$$

  where P represents Precision and R represents Recall (previously calculated).

- *ROC Curve (Receiver Operating Characteristic)*:
  ROC curve is a performance measurement for classification problem at various thresholds settings [19]. If we graphically represent the different levels of sensitivity and specificity of a predictive classification algorithm, we obtain a curve that illustrates how the algorithm performs in terms of precision. This curve is commonly referred to as the *ROC Curve* and also offers insights into the trade-offs between True Positives (TP) and False Positives (FP) rates [26].

  These curves are graphical representations that illustrate how sensitivity changes concerning false positives (which are essentially the complementary of specificity) at different cut-off points. These representations are crucial for determining the most appropriate cutoff point in a test, evaluating its overall performance, and comparing the discriminative ability of two or more algorithmic tests.

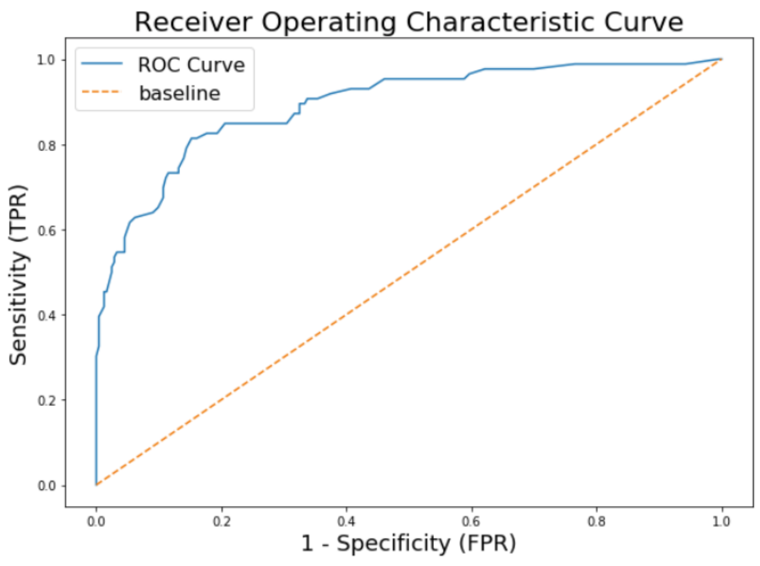

Figure 7: ROC Curve

- *AUC (Area Under Curve)*:
  It is the performance metric associated with the ROC Curve. It can be interpreted as the expected true positive rate and is calculated as follows:
  If we define the function $F_1(t)$ to represent the false positive rate at threshold t on the x-axis, and the function $F_0(t)$ to represent the true positive rate at threshold t on the y-axis, with both functions monotonically non-decreasing as the threshold value t increases, then the Area Under the Curve (AUC) is defined as [10]:

$$\int_0^1 F_0(s) dF_1(s) = \int_{-\infty}^{\infty} F_0(s) f_1(s) ds \tag{10}$$

4 Experimentation

The experimentation was carried out using the following computer: HP 11th Gen Intel(R) Core(TM) i7 2.80GHz 16.0GB RAM.

The programming language used was Python, through the open-source web application called *Jupyter Notebook*, with the following versions of the packages:

Package	**Version**
IPython	7.31.1
ipykernel	6.15.2
ipywidgets	7.6.5
jupyter_client	7.3.4
jupyter_core	4.11.1
jupyter_server	1.18.1
jupyterlab	3.4.4
nbclient	0.5.13
nbconvert	6.4.4
nbformat	5.5.0
notebook	6.4.12
qtconsole	5.2.2
traitlets	5.1.1

Table 4: Jupyter Notebook base packages versions

In this Master's Thesis, the dataset contains one million instances. Since this number is very high, to achieve the proposed objectives, we will select 30% of the data in a stratified manner for both classes. Once the dataset is reduced, there will be 300000 instances, with 296701 belonging to class 0 (non-fraudulent requests) and 3299 classified as fraudulent (class 1), as it can be seen in Figure 8.

Once we have our dataset ready to work on, we perform the corresponding split into training and test sets, respecting the 70% for the training set and 30% for the other set, using a stratified cross-validation technique with a holdout method through the *train_test_split* function from the *sklearn* library.

Next, we will apply the oversampling technique mentioned and explained earlier, known as *SMOTE*, to the training dataset. Specifically, we will use the *SMOTE-NC* technique because we have categorical attributes among the features. This technique is only applied to the training set since it is important for the algorithms to learn as effectively as possible during the training phase with both existing classes.

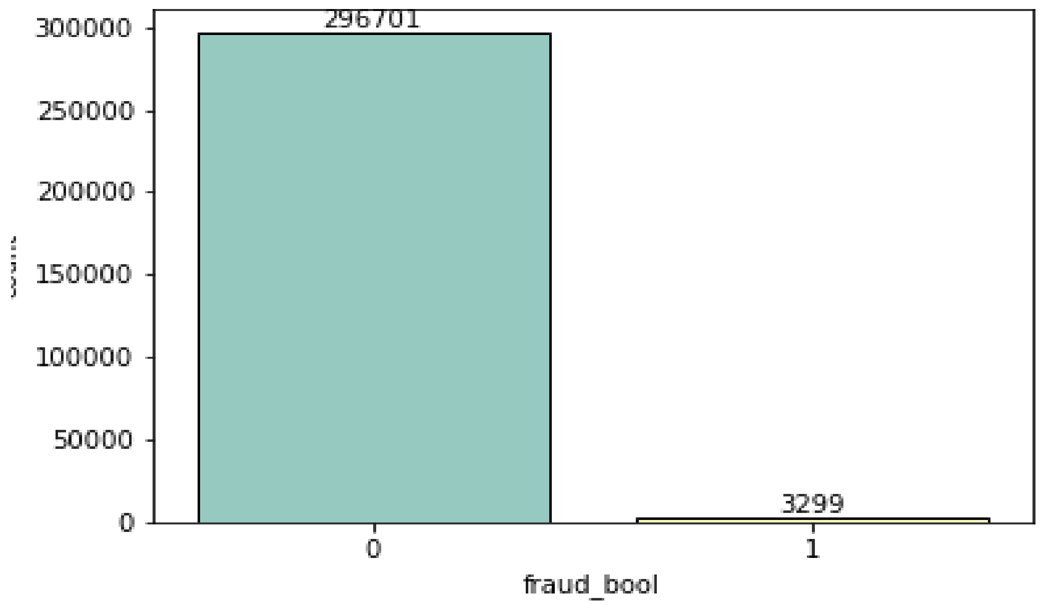

Figure 8: Instances distribution for both classes after choosing 30% of total. *Self-generated image*

Before initiating classifier testing, two pivotal steps must be performed. Firstly, encoding categorical attributes is essential to prevent potential biases in the model, as it could distort the significance of certain attributes during prediction or classification. Secondly, attribute scaling is crucial because some models are sensitive to feature scaling. Inadequate scaling can lead to an imbalance in the impact of attributes with different numerical ranges, potentially resulting in erroneous conclusions. Furthermore, this data preparation approach can enhance computational efficiency by aiding models in achieving convergence more swiftly.

4.1 Algorithm Testing in Python

4.1.1 Logistic Regression

To apply the Logistic Regression algorithm in Python, we will use the *sklearn* library, particularly the function *LogisticRegression* within the *linear_model* submodule.
After using this function and training the algorithm with the *fit* function, predictions are generated, and the resulting outcomes (coefficients and logits) are obtained using the commands *model.coef_* and *np.exp(model.coef_)*, with *np* representing the abbreviation for the imported *numpy* library.

4.1.2 Decision Tree

To apply the Decision Tree algorithm in Python, the following functions have been used:
The function *DecisionTreeClassifier* from the library *sklearn*, with these precise parameters:

- criterion: Function to measure the quality of a split. By defult the function uses the Gini Index, but in this Master's Thesis the Entropy will be used.

- *max_depth*: Maximum depth of the tree. By default, the nodes will expand until all leaves are pure.

- *min_samples_leaf*: Minimum number of samples required to be at a leaf node. Its value by defult is 1.

Once the Decision Tree is created and trained, predictions will be made and the structure can be obtained by using *model.get_depth()* and *model.get_n_leaves()*.

4.1.3 Neural Networks

To work with Neural Networks, the *tensorflow* library will be used, specifically the functions *Sequential* and *layers* from the *keras* submodule. Its structure is as follows: with the *Sequential* function, layers are grouped together, which will be defined using the *layers* function, forming the structure of the Neural Network.

Within the *layers*function, for each of the dense layers of the Neural Network, the *Dense* function is used with the following parameters:

- *activation*: Activation function applied to the output of one of the layers in the network. In this case, the *ReLU (Rectified Linear Unit)* function is chosen.

- *units*: Number of neurons in the layer.

Another function employed, in addition to *Dense*, is the *Dropout* function. It is applied after specific layers in the Neural Network. This function, with a certain probability, deactivates some neurons within a layer during training for a particular instance. This approach helps mitigate overfitting on the training dataset.

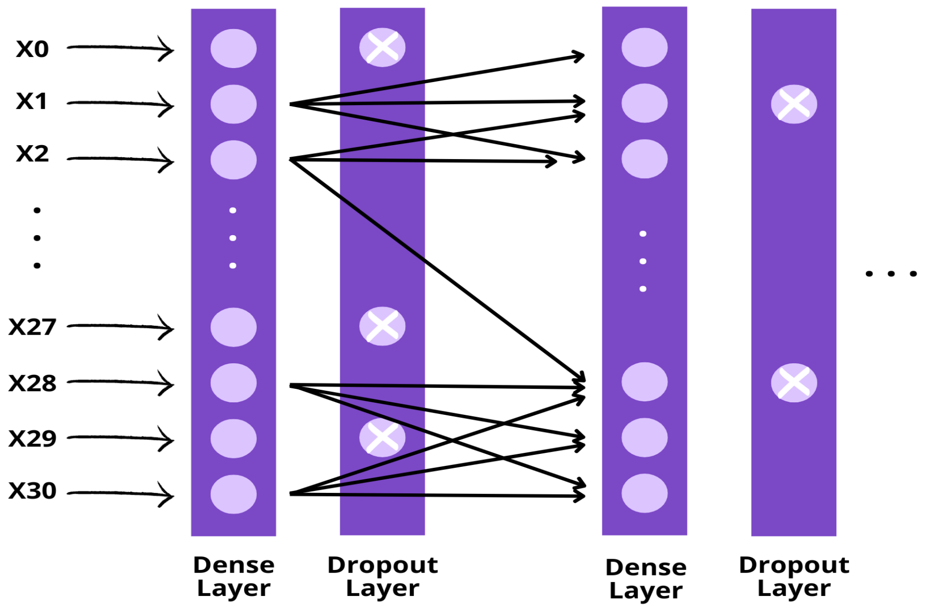

Figure 9: Representation of Dense and Dropout layering. *Self-generated image*

5 Results

5.1 Logistic Regression

The following results were obtained after applying Logistic Regression:

Attribute	**Coeficient**	exp^{coef}
income	0.8120	**2.2524**
name_email_similarity	-1.638	0.1943
prev_address_months_count	-2.910	0.0544
current_address_months_count	0.6339	1.8850
customer_age	**2.4497**	**11.585**
days_since_request	0.1383	1.1484
intended_balcon_account	-0.911	0.4020
payment_type	**0.7392**	**2.0942**
zip_count_4w	0.5929	1.8093
velocity_6h	-0.000	0.9998
velocity_24h	0.4360	1.5466
velocity_4w	0.5870	1.7987
bank_branch_count_8w	-0.296	0.7437
date_of_birth_distinct_emails_4w	-2.766	0.0628
employment_status	-7.466	0.0005
credit_risk_score	**1.0502**	**2.8583**
email_is_free	-0.205	0.8145
housing_status	-5.075	0.0062
phone_home_valid	-3.045	0.0475
phone_mobile_valid	-1.716	0.1797
bank_months_count	0.0593	1.0611
has_other_cards	-3.779	0.0228
proposed_credit_limit	-0.363	0.6950
foreign_request	-1.231	0.2919
source	-4.930	0.0072
session_length_in_minutes	-0.572	0.5640
device_os	**2.7419**	**15.51**
keep_alive_session	-2.084	0.1244
device_distinct_emails_8w	-4.483	0.0112

Table 5: Coefficients of the attributes used in the study

The values in the last column represent the increase in the likelihood of a request being fraudulent as a particular variable increases by one unit. For instance, for every 10-year difference in a customer's age compared to another customer, the probability of fraud increases by 11.585 points. The variables that have the greatest influence on this probability are *device_os, customer_age, credit_risk_score, income* and *payment_type*.

5.1.1 Test Results

Results

		Model	
	Accuracy	88.797	
	Confusion Matrix	**79387**	9623
		459	**531**
	Sensitivity	0.891	
	Specificity	0.536	
	AUC	0.830	
Class 0	**F1-Score**	0.940	
	Precision	0.990	
	Recall	0.890	
Class 1	**F1-Score**	0.100	
	Precision	0.050	
	Recall	0.540	

Table 6: Logistic Regression Results

Based on these results, we can draw the following conclusions:

The model achieves a high accuracy rate, approaching 89%. However, it is important to note that accuracy may not be the most suitable metric for imbalanced datasets, as it does not account for the uneven distribution of classes. Examining the confusion matrix, we can see that class 0 is predominantly classified correctly, while class 1 experiences a high number of misclassifications, resulting in a low recall value of 0.54.

Therefore, with results like these, we can conclude that the Logistic Regression model does not perform the desired function, as the focus should be on correctly classifying fraudulent instances rather than non-fraudulent ones. It is worth noting the very low F1-Score and Precision values for class 1, supporting the earlier observations.

5.2 Decision Tree

To search for an appropriate classifier using Decision Trees, various structures will be proposed. The following models will have different structural constraints imposed on them, such as restricting the tree's depth or setting a minimum number of instances required for a leaf node. These proposed models are:

- **First model**:

  For this initial model, no constraints were applied, and the function's parameters were left at their default values. The resulting tree has a depth of 40 levels and has 7774 leaf nodes. The resulting model has the following feature importances:

Features	Importance
income	0.0260
name_email_similarity	0.0171
prev_address_months_count	0.0035
current_address_months_count	0.0230
customer_age	**0.1389**
days_since_request	0.0108
intended_balcon_account	0.0132
payment_type	0.0090
zip_count_4w	0.0156
velocity_6h	0.0204
velocity_24h	0.0171
velocity_4w	0.0201
bank_branch_count_8w	0.0112
date_of_birth_distinct_emails_4w	0.0088
employment_status	0.0179
credit_risk_score	0.0091
email_is_free	0.0020
housing_status	**0.2731**
phone_home_valid	**0.0843**
phone_mobile_valid	0.0159
bank_months_count	0.0136
has_other_cards	0.0474
proposed_credit_limit	0.0191
foreign_request	0.0006
source	0.0
session_length_in_minutes	0.0098
device_os	**0.0867**
keep_alive_session	**0.0828**
device_distinct_emails_8w	0.0014

Table 7: Feature Importance for the First Decision Tree Model

The attributes that have been given the most importance are *housing_status, customer_age,* *device_os, phone_home_valid* and *keep_alive_*. This means that these attributes are de ones that explain the most (depending on their values) why an instance should be classified as fraudulent or non-fraudulent.

- **<u>Second model</u>**:

  For this model, the constraint of having at least 50 instances in each leaf node has been imposed.
  Here are the feature importances:

Feature	Importance
income	0.0205
name_email_similarity	0.0120
prev_address_months_count	0.0020
current_address_months_count	0.0192
customer_age	**0.1547**
days_since_request	0.0045
intended_balcon_account	0.0059
payment_type	0.0062
zip_count_4w	0.0083
velocity_6h	0.0116
velocity_24h	0.0076
velocity_4w	0.0125
bank_branch_count_8w	0.0053
date_of_birth_distinct_emails_4w	0.0033
employment_status	0.0194
credit_risk_score	0.0029
email_is_free	0.0008
housing_status	**0.3108**
phone_home_valid	**0.0962**
phone_mobile_valid	0.0170
bank_months_count	0.0094
has_other_cards	0.0542
proposed_credit_limit	0.0178
foreign_request	0.0000
source	0.0000
session_length_in_minutes	0.0043
device_os	**0.0978**
keep_alive_session	**0.0945**
device_distinct_emails_8w	0.000

Table 8: Feature Importance for the Second Decision Tree Model

In this case, the model has a depth of 29 levels and 2144 leaf nodes, and the most important features are: *housing_status, customer_age, device_os, phone_home_valid* and *keep_alive_session.*

- **Third model**:

  For this last case, a more restrictive model has been created with a maximum tree depth of 10 and a minimum of 1500 instances per leaf.
  The results are:

Feature	**Importance**
income	0.0081
name_email_similarity	0.0069
prev_address_months_count	0.0008
current_address_months_count	0.0158
customer_age	**0.1451**
days_since_request	0.0010
intended_balcon_account	0.0006
payment_type	0.0013
zip_count_4w	0.0036
velocity_6h	0.0061
velocity_24h	0.0035
velocity_4w	0.0065
bank_branch_count_8w	0.0021
date_of_birth_distinct_emails_4w	0.0010
employment_status	0.0190
credit_risk_score	0.0006
email_is_free	0.0008
housing_status	**0.3557**
phone_home_valid	**0.1101**
phone_mobile_valid	0.0159
bank_months_count	0.0024
has_other_cards	0.0620
proposed_credit_limit	0.0090
foreign_request	0.0000
source	0.0000
session_length_in_minutes	0.0006
device_os	**0.1118**
keep_alive_session	**0.1083**
device_distinct_emails_8w	0.0000

  Table 9: Feature Importance for the Third Decision Tree Model

  A tree with a depth of 10 and 540 leaf nodes was obtained in this case. The variables with the highest importance for predicting other values were: *housing_status, customer_age, device_os, phone_home_valid* and *keep_alive_session.*

5.2.1 Test Results

		Results					
		First Model		*Second Model*		*Third Model*	
	Depth	40		29		10	
	Leaf Nodes	7774		2144		540	
	Accuracy	95.821		94.977		92.117	
	Confusion Matrix	**86028**	2982	**85197**	3813	**82459**	6551
		779	**211**	707	**283**	543	**447**
	Sensitivity	0.966		0.957		0.926	
	Specificity	0.213		0.285		0.451	
	AUC	0.589		0.767		0.805	
Class 0	**F1-Score**	0.980		0.970		0.960	
	Precision	0.990		0.990		0.990	
	Recall	0.970		0.960		0.930	
Class 1	**F1-Score**	0.100		0.110		0.110	
	Precision	0.070		0.070		0.060	
	Recall	0.210		0.290		0.450	

Table 10: Decision Trees Test Results

The three models yield highly similar results, all achieving accuracy rates above 90%. However, it is important to note that relying solely on accuracy may be misleading in this context. A closer examination of the confusion matrix reveals that the first two models perform poorly in classifying instances belonging to class 1, with the third model demonstrating a slight improvement, though still classifying fewer than half of them correctly.

This limitation is further evident when assessing the specificity metric, as none of the three models even reach a value of 0.5.

Comparing these models, the third one emerges as the preferred choice. It not only delivers better results in predicting the positive class but also offers a simpler and more interpretable structure due to its shallower depth.

5.3 Neural Networks

For this algorithm, several structures have been proposed to evaluate how the results vary with different designs. These structures are:

- *15-30-10-2 with dropout layers*:
  The network consists of a total of 4 layers, with 15, 30, 10, and 2 neurons in each, respectively. Behind each layer of neurons, there will be a dropout layer with a probability of 0.1 for deactivating a neuron. Its representation is:
- *15-30-10-2 without dropout layers*:
  Same structure of neuron layers as before, but they are not followed by a dropout layer. Its graphical representation is:
- *30-15-2 with dropout layers*:
  This network has one fewer layer and now has 30, 15, and 2 neurons in each layer, respectively. This network does have a dropout layer behind each neuron, with a probability of 0.1.

5.3.1 Test Results

The three models that have been explained and trained before have the following results on the test set:

		Neural Networks					
		First Model		*Second Model*		*Third Model*	
	Structure	*15-30-10-2 (with dropout)*		*15-30-10-2 (w/o dropout)*		*30-15-2 (with dropout)*	
	Accuracy	88.152		98.800		88.610	
	Confusion Matrix	**78801**	10218	**88967**	0	**79162**	9805
		445	**536**	1033	**0**	447	**586**
	Sensitivity	0.885		1.000		0.889	
	Specificity	0.546		0.000		0.567	
	AUC	0.715		0.500		0.728	
Class 0	**F1-Score**	0.940		0.990		0.940	
	Precision	0.990		0.990		0.990	
	Recall	0.890		1.000		0.890	
Class 1	**F1-Score**	0.090		0.000		0.100	
	Precision	0.050		0.000		0.060	
	Recall	0.550		0.000		0.530	

Table 11: Neural Network Test Results

Table 11 reveals:

The results might appear very promising when considering the accuracy levels, especially the second proposed model, which achieves nearly 100% accuracy in classifying instances. However, as with the previous algorithms, the accuracy metric can be misleading about the actual performance of the models. Despite the high accuracy values, other metrics demonstrate that the prediction accuracy for fraudulent instances is only around 50%.

In the second model, this prediction accuracy is extremely poor, as evident from the confusion matrix, where the network has been trained to classify all instances as non-fraudulent, resulting in a high overall accuracy rate. Clearly, the second model is not a viable choice.

The other two models yield similar results across all evaluation metrics, with comparable structures, differing only in that the third model has one fewer hidden layer. The execution times for these models were 827.006 seconds for the first model and 862.699 seconds for the second.

To conclude the Neural Network algorithm, the optimal model would be the third one, featuring a sequence of layers consisting of 30, 15, and 2 neurons each, along with dropout layers with a probability of 0.1 following each neuron layer.

6 Conclusions

For this Master's Thesis, two main goals have been proposed.

The first goal, which involves the proper preparation and data treatment, has been addressed after conducting a descriptive study and applying data rebalancing and reduction techniques. The result was a dataset that was split into training and testing sets, suitable for use in classification algorithms. Oversampling techniques were applied to the training set to obtain a balanced dataset that accurately represents both existing classes. A balanced dataset helps the algorithm train more effectively and learn to differentiate the minority class more easily.

Once the dataset has been meticulously prepared to facilitate the application of the designated algorithms, the goal becomes apparent: to discern which of the chosen classification algorithms is the most effective in solving the problem while giving paramount importance to the accurate classification of fraudulent instances.

Three distinct algorithms have been harnessed for this task: Logistic Regression, Decision Trees, and Neural Networks. Notably, Logistic Regression and Decision Trees exhibit relatively low execution times, typically measured in seconds.
Upon evaluating the selected algorithms and their respective configurations, it becomes evident that they yield similar and praiseworthy results when it comes to classifying non-fraudulent instances. Nevertheless, it's worth reiterating that the pivotal concern lies in correctly identifying fraudulent instances. Regrettably, these algorithms often falter in this regard, misclassifying them in approximately half of the cases during testing.

References

[1] Herve Abdi, Dominique Valentin, and Betty Edelman. *Neural networks.* Number 124. Sage, 1999.

[2] Ethem Alpaydin. *Machine learning.* MIT Press, 2021.

[3] Taiwo Oladipupo Ayodele. Types of machine learning algorithms. *New advances in machine learning*, 3:19–48, 2010.

[4] Raquel Beltrán Barba. Modelos de Aprendizaje Profundo. Aplicaciones con R y Python. Master's thesis, University of Seville, 2022.

[5] J. Martin Bland and Douglas G. Altman. The odds ratio. *BM*, 320(7247):1468, 2000.

[6] Giuseppe Bonaccorso. *Machine learning algorithms.* Packt Publishing Ltd, 2017.

[7] Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research 16*, pages 321—-357, 2011.

[8] Bahzad Charbuty and Adnan Abdulazeez. Classification based on decision tree algorithm for machine learning. *Journal of Applied Science and Technology Trends*, 2(01):20–28, 2021.

[9] Hercules Dalianis. *Evaluation Metrics and Evaluation*, pages 45–53. Springer International Publishing, Cham, 2018.

[10] César Ferri, José Hernández-Orallo, and Peter A. Flach. A coherent interpretation of AUC as a measure of aggregated classification performance. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 657–664, 2011.

[11] Jeff Heaton. *Artificial intelligence for humans.* Heaton Research, Inc., 2015.

[12] Joseph M. Hilbe. *Logistic regression models.* CRC Press, 2009.

[13] T. Ryan Hoens and Nitesh V. Chawla. *Imbalanced Datasets: From Sampling to Classifiers.* John Wiley Sons, Ltd, 2013.

[14] Mohammad Hossin and Md Nasir Sulaiman. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process (IJDKP)*, 5(2):1, 2015.

[15] Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. *Electronic Markets*, 31(3):685–695, 2021.

[16] Sérgio Jesus, José Pombal, Duarte Alves, André Cruz, Pedro Saleiro, Rita P. Ribeiro, João Gama, and Pedro Bizarro. Turning the Tables: Biased, Imbalanced, Dynamic Tabular Datasets for ML Evaluation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[17] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research*, 18(1):559–563, 2017.

[18] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9(1):381–386, 2020.

[19] Sarang Narkhede. Understanding auc-roc curve. *Towards Data Science*, 26:220–227, 2018.

[20] Anuja Priyam, Gupta R. Abhijeeta, Anju Rathee, and Saurabh Srivastava. Comparative analysis of decision tree classification algorithms. *International Journal of current engineering and technology*, 3, 2013.

[21] Shagan Sah. Machine Learning: A Review of Learning Types. *Preprints*, July 2020.

[22] Paola Santana Morales. Aplicación de técnicas múltiples de minería de datos para toma de decisiones en un caso real de gestión de reclamaciones de una compañía de seguros. Master's thesis, University of Huelva, 2022.

[23] Himani Sharma, Sunil Kumar, et al. A survey on decision tree algorithms of classification in data mining. *International Journal of Science and Research (IJSR)*, 5(4):2094–2097, 2016.

[24] Suryakanthi Tangirala. Evaluating the impact of gini index and information gain on classification using decision tree classifier algorithm. *International Journal of Advanced Computer Science and Applications*, 11, 2020.

[25] Steven Tenny and Mary R. Hoffman. *Odds Ratio*. StatPearls Publishing, Treasure Island (FL), 2022.

[26] Ž. Vujović et al. Classification model evaluation metrics. *International Journal of Advanced Computer Science and Applications*, 12(6):599–606, 2021.

[27] Zhi-Hua Zhou. *Machine learning*. Springer Nature, 2021.

A Tables

payment_type

Value	*Number of occurrences*
AB	370554
AA	258249
AC	252071
AD	118837
AE	289

Table 12: Values that attribute payment_type can take

employment_status

Value	*Number of aparitions*
CA	730252
CB	138288
CF	44034
CC	377758
CD	26522
CE	22693
CG	453

Table 13: Values that attribute employment_status can take

housing_status

Value	*Number of aparitions*
BC	372143
BB	260965
BA	169675
BE	169135
BD	26161
BF	1669
BG	252

Table 14: Values that attribute housing_status can take

device_os

Value	*Number of aparitions*
other	342728
linux	332712
windows	263506
macintosh	53826
x11	7228

Table 15: Values that attribute device_os can take

source

Value	*Number of aparitions*
INTERNET	992952
APP	7048

Table 16: Values that attribute source can take

	count	unique	top	freq	mean	std	min	25 %	50 %	75 %	max
fraud_bool	1000000	NaN	NaN	NaN	0.011029	0.104438	0.000000	0.000000	0.000000	0.000000	1.000000
income	1000000	NaN	NaN	NaN	0.562696	0.290343	0.100000	0.300000	0.600000	0.800000	0.900000
name_email_similarity	1000000	NaN	NaN	NaN	0.493694	0.289125	0.000001	0.225216	0.492153	0.755567	0.999999
prev_address_months_count	1000000	NaN	NaN	NaN	16.718568	44.046230	-1.000000	-1.000000	-1.000000	12.000000	383.000000
current_address_months_count	1000000	NaN	NaN	NaN	86.587867	88.406599	-1.000000	19.000000	52.000000	130.000000	428.000000
customer_age	1000000	NaN	NaN	NaN	33.689080	12.025799	10.000000	20.000000	30.000000	40.000000	90.000000
days_since_request	1000000	NaN	NaN	NaN	1.025705e+00	5.381835e+00	4.036860e-09	7.193246e-03	1.517574e-02	2.633069e-02	7.845690e+01
intended_balcon_amount	1000000	NaN	NaN	NaN	8.661499	20.236155	-15.530555	-1.181488	-0.830507	4.984176	112.956928
payment_type	1000000	5	AB	370554	NaN	NaN	NaN	NaN	NaN	NaN	NaN
zip_count_4w	1000000	NaN	NaN	NaN	1572.692049	1005.374565	1.000000	894.000000	1263.000000	1944.000000	6700.000000
velocity_6h	1000000	NaN	NaN	NaN	5665.296605	3009.380665	-170.603072	3436.365848	5319.769349	7680.717827	16715.565404
velocity_24h	1000000	NaN	NaN	NaN	4769.781965	1479.212612	1300.307314	3593.179135	4749.921161	5752.574191	9506.896596
velocity_4w	1000000	NaN	NaN	NaN	4856.324016	919.843934	2825.748405	4268.368423	4913.436941	5488.083356	6994.764201
bank_branch_count_8w	1000000	NaN	NaN	NaN	184.361849	459.625329	0.000000	1.000000	9.000000	25.000000	2385.000000
date_of_birth_distinct_emails_4w	1000000	NaN	NaN	NaN	9.503544	5.033792	0.000000	6.000000	9.000000	13.000000	39.000000
employment_status	1000000	7	CA	730252	NaN	NaN	NaN	NaN	NaN	NaN	NaN
credit_risk_score	1000000	NaN	NaN	NaN	130.989595	69.681812	-170.000000	83.000000	122.000000	178.000000	389.000000
email_is_free	1000000	NaN	NaN	NaN	0.529886	0.499106	0.000000	0.000000	1.000000	1.000000	1.000000
housing_status	1000000	7	BC	372143	NaN	NaN	NaN	NaN	NaN	NaN	NaN
phone_home_valid	1000000	NaN	NaN	NaN	0.417077	0.493076	0.000000	0.000000	0.000000	1.000000	1.000000
phone_mobile_valid	1000000	NaN	NaN	NaN	0.889676	0.313293	0.000000	1.000000	1.000000	1.000000	1.000000
bank_months_count	1000000	NaN	NaN	NaN	10.839303	12.116875	-1.000000	-1.000000	5.000000	25.000000	32.000000
has_other_cards	1000000	NaN	NaN	NaN	0.222988	0.416251	0.000000	0.000000	0.000000	0.000000	1.000000
proposed_credit_limit	1000000	NaN	NaN	NaN	515.851010	487.559902	190.000000	200.000000	200.000000	500.000000	2100.000000
foreign_request	1000000	NaN	NaN	NaN	0.025242	0.156859	0.000000	0.000000	0.000000	0.000000	1.000000
source	1000000	2	INTERNET	992952	NaN	NaN	NaN	NaN	NaN	NaN	NaN
session_length_in_minutes	1000000	NaN	NaN	NaN	7.544940	8.033106	-1.000000	3.103053	5.114321	8.866131	85.899143
device_os	1000000	5	other	342728	NaN	NaN	NaN	NaN	NaN	NaN	NaN
keep_alive_session	1000000	NaN	NaN	NaN	0.576947	0.494044	0.000000	0.000000	1.000000	1.000000	1.000000
device_distinct_emails_8w	1000000	NaN	NaN	NaN	1.018312	0.180761	-1.000000	1.000000	1.000000	1.000000	2.000000
device_fraud_count	1000000	NaN	NaN	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.0
month	1000000	NaN	NaN	NaN	3.288674	2.209994	0.000000	1.000000	3.000000	5.000000	7.000000

Table 17: Summary of the dataset attributes

B Programming code

```
"""
Libraries
"""

import pandas as pd
import numpy as np
import sklearn

from collections import Counter
from imblearn.over_sampling import SMOTENC
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
import imblearn
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn import linear_model
import tensorflow as tf
from tensorflow import keras
from keras import layers
from sklearn.metrics import classification_report, confusion_matrix,
                            accuracy_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

################################################################################
                            #READ DATASET
################################################################################
data_raw = pd.read_csv("Base.csv")
data_raw.head()

# Shuffling instancies
data_shu = sklearn.utils.shuffle(data_raw)
data_shu
data_shu.info()

#Distribution for each class
data_shu["fraud_bool"].value_counts() #Datos desbalanceados

#Deleting irrelevant features
datos_pre = data_shu.drop(['month', 'device_fraud_count'], axis=1)

# Heatmap
plt.figure(figsize = (25, 20))
sns.heatmap(datos_pre.corr(),
            annot = True,
            cmap = "Blues",
            fmt = ".2f",
            vmin = -1.00, vmax = 1.00)
plt.savefig("mapa_calor_prev3")

```

```

################################################################################
                           #INSTANCES REDUCTION
################################################################################
y_datos = datos_pre.iloc[:,0]
X_datos = datos_pre.iloc[:,range(1,30)]

X_b, X_g, y_b, y_g = train_test_split(X_datos, y_datos, test_size=0.30,
                                       random_state=42)
(X_g.shape, y_g.shape)
y_g.value_counts()

    #X_g and y_g are the data we will work with



################################################################################
                        # TRAIN/TEST SPLIT
################################################################################
# Categorical features codification
categoricas = ["payment_type", "employment_status", "email_is_free",
               "housing_status", "phone_home_valid", "phone_mobile_valid",
               "has_other_cards", "foreign_request", "source", "device_os",
               "keep_alive_session"]

encoder = LabelEncoder()
for i in categoricas:
    X_g[i] = encoder.fit_transform(X_g[i].astype('object'))

X_g.describe()

# 70/30 split
X_train, X_test, y_train, y_test = train_test_split(X_g, y_g, test_size=0.30,
                                                    random_state=42)

X_train.info()
X_train.shape

X_test.info()
X_test.shape

y_train.shape
y_test.shape

# Class distribution for training set
pentr = pd.concat([X_train, y_train], axis = 1)
ax21 = sns.countplot(x=pentr["fraud_bool"], palette="Set3",
                     edgecolor = "black")
for label in ax21.containers:
    ax21.bar_label(label)
plt.savefig("clase_entrv3")



```

```

###############################################################################
                                        #SMOTE
###############################################################################

print('Original dataset shape %s' % Counter(y_train))

sm = SMOTENC(categorical_features=[7,14,17,24,26]) #indico variables categoricas
X_train, y_train = sm.fit_resample(X_train, y_train)

print('Resampled dataset shape %s' % Counter(y_train)) #Now its balanced

y_train.shape
X_train.shape
data_smote = pd.concat([X_train, y_train], axis = 1)
data_smote.describe()


ax2 = sns.countplot(x=data_smote["fraud_bool"], palette="Set3",
                    edgecolor = "black")
for label in ax2.containers:
    ax2.bar_label(label)
plt.savefig("clase_smotev3")

    # Scaling attributes
escaladorX = MinMaxScaler()
X_train = escaladorX.fit_transform(X_train.astype('float64'))
X_test  = escaladorX.transform(X_test.astype('float64'))

###############################################################################
                        # LOGISTIC REGRESSION
###############################################################################
modelo1 = linear_model.LogisticRegression(max_iter=1000)
modelo1.fit(X_train,y_train)

# Test predictions
y_pred = modelo1.predict(X_test)

modelo1.coef_ #coeficients
np.exp(modelo1.coef_)

#Resultados sobre el conjunto test
modelo1.score(X_test,y_test)
confusion_matrix(y_test, y_pred)
print(classification_report(y_test, y_pred))

# ROC Curve
prob = modelo1.predict_proba(X_test)
probs = prob[:,1]
fpr, tpr, thresholds = roc_curve(y_test, probs)

def plot_roc_curve(fper, tper):
    plt.plot(fper, tper, color='red', label='ROC')
    plt.plot([0, 1], [0, 1], color='green', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic Curve')
    plt.legend()
    plt.show()
plot_roc_curve(fpr, tpr)

auc = roc_auc_score(y_test, probs)
print('AUC: %.2f' % auc)


###############################################################################
                        # DECISION TREES
###############################################################################

### FIRST MODEL
modelo2 = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
modelo2.fit(X_train, y_train)

print(f"Tree depth: {modelo2.get_depth()}")
print(f"Leaf nodes: {modelo2.get_n_leaves()}")


plot2 = plot_tree(
            decision_tree = modelo2,
            class_names   = 'Fraudbool',
            filled        = True,
            impurity      = False,
            fontsize      = 10,
            precision     = 2)

# Test predictions
y_pred2 = modelo2.predict(X_test)

modelo2.score(X_test,y_test)
confusion_matrix(y_test, y_pred2)
print(classification_report(y_test, y_pred2))

# ROC Curve
prob2 = modelo2.predict_proba(X_test)
probs2 = prob2[:,1]
fpr2, tpr2, thresholds2 = roc_curve(y_test, probs2)

def plot_roc_curve(fper, tper):
    plt.plot(fper, tper, color='red', label='ROC')
    plt.plot([0, 1], [0, 1], color='green', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic Curve')
    plt.legend()
    plt.show()
plot_roc_curve(fpr2, tpr2)

auc2 = roc_auc_score(y_test, probs2)
```

```
print('AUC: %.2f' % auc2)



### Segundo modelo (minimo 50 instancias por hoja final)
modelo21 = DecisionTreeClassifier(criterion = 'entropy', random_state = 0,
                                  min_samples_leaf=50)
modelo21.fit(X_train, y_train)

print(f"Profundidad del  rbol : {modelo21.get_depth()}")
print(f"N mero de nodos terminales: {modelo21.get_n_leaves()}")

#Representacion grafica del arbol
plot21 = plot_tree(
            decision_tree = modelo21,
            class_names   = 'Fraudbool',
            filled        = True,
            impurity      = False,
            fontsize      = 10,
            precision     = 2)

#Predicciones sobre el conjunto test
y_pred21 = modelo21.predict(X_test)

modelo21.score(X_test,y_test)
confusion_matrix(y_test, y_pred21)
print(classification_report(y_test, y_pred21))

    # Curva ROC
prob21 = modelo21.predict_proba(X_test)
probs21 = prob21[:,1]
fpr21, tpr21, thresholds21 = roc_curve(y_test, probs21)

def plot_roc_curve(fper, tper):
    plt.plot(fper, tper, color='red', label='ROC')
    plt.plot([0, 1], [0, 1], color='green', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic Curve')
    plt.legend()
    plt.show()
plot_roc_curve(fpr21, tpr21)

auc21 = roc_auc_score(y_test, probs21)
print('AUC: %.2f' % auc21)




### Tercer modelo (minimo 1500 instancias por hoja y maxima profundidad 10)
modelo22 = DecisionTreeClassifier(criterion = 'entropy', random_state = 0,
                                  min_samples_leaf=50, max_depth=10)
modelo22.fit(X_train, y_train)

```

```
print(f"Profundidad del  rbol : {modelo22.get_depth()}")
print(f"N mero de nodos terminales: {modelo22.get_n_leaves()}")

#Representacion grafica del arbol
plot22 = plot_tree(
            decision_tree = modelo22,
            class_names   = 'Fraudbool',
            filled        = True,
            impurity      = False,
            fontsize      = 10,
            precision     = 2)

#Predicciones sobre el conjunto test
y_pred22 = modelo22.predict(X_test)

modelo22.score(X_test,y_test)
confusion_matrix(y_test, y_pred22)
print(classification_report(y_test, y_pred22))

    # Curva ROC
prob22 = modelo22.predict_proba(X_test)
probs22 = prob22[:,1]
fpr22, tpr22, thresholds22 = roc_curve(y_test, probs22)

def plot_roc_curve(fper, tper):
    plt.plot(fper, tper, color='red', label='ROC')
    plt.plot([0, 1], [0, 1], color='green', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic Curve')
    plt.legend()
    plt.show()
plot_roc_curve(fpr22, tpr22)

auc22 = roc_auc_score(y_test, probs22)
print('AUC: %.2f' % auc22)


################################################################################
                                # REDES NEURONALES
################################################################################
#Hiperparamteros predefinidos
TAM_BATCH = 32
TAM_SHUFFLE = 1000
LOSS_FUNC = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
ACC_FUNC = tf.keras.metrics.SparseCategoricalAccuracy()

EPOCAS = 50
L_RATE= 0.0001
OPTIM = tf.keras.optimizers.SGD(learning_rate=L_RATE)


### FIRST MODEL
    #Structure
```

```
modelo1 = keras.Sequential([
    layers.Dense(15, activation = "relu"),
    tf.keras.layers.Dropout(0.1),
    layers.Dense(30, activation = "relu"),
    tf.keras.layers.Dropout(0.1),
    layers.Dense(10, activation = "relu"),
    tf.keras.layers.Dropout(0.1),
    layers.Dense(2, activation = "relu"),
])

modelo1.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.0001),
              loss=LOSS_FUNC,
              metrics=ACC_FUNC)

history21 = modelo1.fit(X_train, y_train, epochs=EPOCAS)

    #Accuracy and loss representation
plt.figure()
plt.plot(history21.history['loss'], label='Loss (training data)')
plt.title('Loss for Bank Fraud Dataset')
plt.ylabel('CE')
plt.xlabel('Epoch')
plt.legend(loc="upper right")
plt.show()

# Plot history: Accuracy
plt.figure()
plt.plot(history21.history['sparse_categorical_accuracy'],
        label='Accuracy (training data)')
plt.title('accuracy for Bank Fraud Dataset')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(loc="upper left")
plt.show()

    #Test results
modelo1.evaluate(X_test,  y_test)
y_pred31 = modelo1.predict(X_test)

print('Accuracy: %.5f' % accuracy_score(y_test, np.argmax(y_pred31, axis=1 )))
print(classification_report(y_test, np.argmax(y_pred31, axis=1 )))

confusion_matrix = confusion_matrix(y_test, np.argmax(y_pred31, axis=1 ))
print(confusion_matrix)




### SECOND MODEL
    #Structure
modelo2 = keras.Sequential([
    layers.Dense(15, activation = "relu"),
    layers.Dense(30, activation = "relu"),
    layers.Dense(10, activation = "relu"),
    layers.Dense(2, activation = "relu"),
])

modelo2.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.0001),
              loss=LOSS_FUNC,
              metrics=ACC_FUNC)

history22 = modelo1.fit(X_train, y_train, epochs=EPOCAS)

    #Accuracy and loss representation
plt.figure()
plt.plot(history22.history['loss'], label='Loss (training data)')
plt.title('Loss for Bank Fraud Dataset')
plt.ylabel('CE')
plt.xlabel('Epoch')
plt.legend(loc="upper right")
plt.show()

# Plot history: Accuracy
plt.figure()
plt.plot(history22.history['sparse_categorical_accuracy'],
        label='Accuracy (training data)')
plt.title('accuracy for Bank Fraud Dataset')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(loc="upper left")
plt.show()

    #Test results
modelo2.evaluate(X_test,  y_test)
y_pred32 = modelo2.predict(X_test)

print('Accuracy: %.5f' % accuracy_score(y_test, np.argmax(y_pred32, axis=1 )))
print(classification_report(y_test, np.argmax(y_pred32, axis=1 )))

confusion_matrix2 = confusion_matrix(y_test, np.argmax(y_pred32, axis=1 ))
print(confusion_matrix2)



### THIRD MODEL
    #Structure
modelo3 = keras.Sequential([
    layers.Dense(64, activation = "relu"),
    tf.keras.layers.Dropout(0.1),
    layers.Dense(128, activation = "relu"),
    tf.keras.layers.Dropout(0.1),
    layers.Dense(256, activation = "relu"),
    tf.keras.layers.Dropout(0.1),
    layers.Dense(512, activation = "relu"),
    tf.keras.layers.Dropout(0.1),
    layers.Dense(256, activation = "relu"),
    tf.keras.layers.Dropout(0.1),
    layers.Dense(128, activation = "relu"),
    tf.keras.layers.Dropout(0.1),
    layers.Dense(2, activation = "relu"),
])

modelo3.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.0001),
              loss=LOSS_FUNC,
              metrics=ACC_FUNC)

history23 = modelo1.fit(X_train, y_train, epochs=EPOCAS)

    #Accuracy and loss representation
plt.figure()
plt.plot(history23.history['loss'], label='Loss (training data)')
plt.title('Loss for Bank Fraud Dataset')
plt.ylabel('CE')
plt.xlabel('Epoch')
plt.legend(loc="upper right")
plt.show()

# Plot history: Accuracy
plt.figure()
plt.plot(history23.history['sparse_categorical_accuracy'],
        label='Accuracy (training data)')
plt.title('accuracy for Bank Fraud Dataset')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(loc="upper left")
plt.show()

    #Test results
modelo3.evaluate(X_test,  y_test)
y_pred33 = modelo3.predict(X_test)

print('Accuracy: %.5f' % accuracy_score(y_test, np.argmax(y_pred33, axis=1 )))
print(classification_report(y_test, np.argmax(y_pred33, axis=1 )))

confusion_matrix3 = confusion_matrix(y_test, np.argmax(y_pred33, axis=1 ))
print(confusion_matrix3)
```