**Ministry of Science and Higher Education of the Republic of Kazakhstan**

**L.N. Gumilyov Eurasian National University**

**Faculty of Information Technology**

**Department of Information Systems**

**COURSEWORK**

ON THE SUBJECT

"Mathematical Foundations of Intelligent Systems"

For third-year students of the specialty 6B06103 – Information Systems

Topic: **Fraud detection in financial transactions using machine learning algorithms.**

**Completed by:**

Student of group IS-32

Suleimenova Inkara

**Coursework Supervisor:**

Prof., Zhukabayeva T. K.

Full name, signature

**Members of the Commission:**

Assoc. Prof. Muhanova А.А.

Full name, signature

PhD, Serikbayeva S.K.

Full name, signature

Full name, signature

Grade   
« » 2024

**Astana, 2024**

**Table of content**

[Introduction 3](#_Toc184714412)

[The Role of Machine Learning Algorithms 5](#_Toc184714413)

[1.1. Fraud detection and machine learning. 5](#_Toc184714414)

[Dataset and Preprocessing 7](#_Toc184714415)

[2.1 Exploratory Data Analysis 7](#_Toc184714416)

[2.2 Data Description 7](#_Toc184714417)

[2.3 Data Prossesing 9](#_Toc184714418)

[Client application creation 24](#_Toc184714419)

[3.1 Database implementation 24](#_Toc184714420)

[3.2 Client application creation 29](#_Toc184714421)

[3.3 Machine learning implementation 41](#_Toc184714422)

[Conclusion 44](#_Toc184714423)

[Reference 45](#_Toc184714424)

# **Introduction**

Fraud detection in financial transactions has grown increasingly critical as the volume and complexity of digital transactions continue to expand. This digital evolution introduces new vulnerabilities and sophisticated methods of fraud that financial institutions must address. The stakes are high—fraudulent activities can result in significant financial losses, reputational harm, and severe security breaches. The primary objective of this project is to design a robust framework for detecting fraudulent transactions using advanced machine learning algorithms, enabling the identification of suspicious activities in real time and reducing false positives.

Fraud detection in the realm of intelligent systems and machine learning presents a complex problem due to the dynamic and adaptive nature of fraudulent behaviors. Fraudsters continuously evolve their strategies, creating challenges for traditional detection mechanisms. Furthermore, the imbalanced nature of transaction data—where fraudulent transactions constitute a small percentage compared to genuine ones—complicates model training and evaluation. Additionally, the need for high-speed, accurate decision-making underlines the importance of scalable and efficient machine learning solutions. Addressing these challenges requires leveraging large datasets to uncover hidden patterns and subtle anomalies indicative of fraudulent activity.

This project explores various mathematical and computational models, including logistic regression, decision trees, random forests, and neural networks. Each algorithm offers unique benefits:

* Logistic Regression: Provides a probabilistic framework for binary classification, efficiently separating legitimate and fraudulent transactions.
* Decision Trees: Allow intuitive, rule-based predictions that are easy to interpret, making them suitable for initial exploratory analysis.
* Random Forests: Combine the predictive power of multiple decision trees, improving accuracy and robustness.
* Neural Networks: Use layered structures to learn complex, non-linear patterns in high-dimensional data, making them particularly effective for detecting subtle fraudulent behavior.

# **The Role of Machine Learning Algorithms**

## Fraud detection and machine learning.

The financial sector increasingly relies on machine learning to enhance security and mitigate risks. Innovations in anomaly detection, predictive modeling, and pattern recognition are vital to improving the reliability and integrity of financial systems. According to industry reports, cybercrime is projected to cost the global economy over $9.5 trillion USD in 2024, underscoring the urgency of adopting advanced detection systems. For instance, Citibank reported a 70% reduction in phishing attacks through the application of machine learning, while Walmart decreased shoplifting by 25% using real-time video analysis. These examples demonstrate the transformative potential of machine learning across diverse applications.

The advantages of integrating machine learning into fraud detection workflows include:

* Faster and More Accurate Detection: Machines excel at processing large datasets, identifying suspicious patterns far faster than manual methods.
* Reduced Manual Effort: Automating the detection process frees up resources and minimizes the time spent on reviewing transactions manually.
* Scalability: Machine learning models can handle fluctuating transaction volumes, making them ideal for businesses experiencing seasonal variations.
* Adaptability: Algorithms learn from new data, ensuring that detection systems stay ahead of evolving fraud techniques.
* Cost Efficiency: Once implemented, machine learning systems reduce reliance on human resources for fraud monitoring, providing a scalable solution with lower operational costs.

Fraud detection is a cornerstone of financial security, with applications spanning banking, e-commerce, insurance, and government sectors. Machine learning-powered systems can:

* Prevent financial losses by identifying suspicious transactions before they occur.
* Enhance customer trust by ensuring secure transactions and protecting sensitive information.
* Support compliance efforts by adhering to anti-fraud regulations and avoiding penalties.
* Streamline operations by reducing delays caused by fraudulent transactions.
* Mitigate global economic risks associated with large-scale financial fraud.

# **Dataset and Preprocessing**

The dataset used in this project is the "Synthetic Financial Datasets for Fraud Detection" available on Kaggle. It simulates mobile money transactions and incorporates malicious behaviors for testing fraud detection methods. The dataset contains over 6 million entries, reduced to 2.8 million after retaining only the relevant transaction types—CASH-OUT and TRANSFER. These transaction types were identified as the primary categories associated with fraudulent activity.

## Exploratory Data Analysis

Preliminary analysis revealed a significant class imbalance:

* Total transactions: 6,362,620
* Fraudulent transactions: 8,213 (0.13%)
* Legitimate transactions: 6,354,407 (99.87%)

The imbalance highlights the need for techniques such as oversampling, undersampling, or synthetic data generation to ensure that models effectively learn to detect fraudulent transactions.

## Data Description

The dataset utilized in this analysis was synthetically created using the PaySim simulator, which replicates digital financial transactions. PaySim operates by modeling mobile money transactions based on anonymized samples of real financial logs collected over a month from a mobile money service in an African country. By aggregating data from private records, the simulator generates synthetic transaction data and incorporates fraudulent activities for evaluation purposes. This dataset comprises over 6 million records and 11 features. One crucial column, labeled 'isFraud,' identifies whether a transaction is genuinely fraudulent, serving as the target variable for our analysis. Below is a breakdown of the dataset's key columns and their descriptions.

1,PAYMENT,1060.31,C429214117,1089.0,28.69,M1591654462,0.0,0.0,0,0

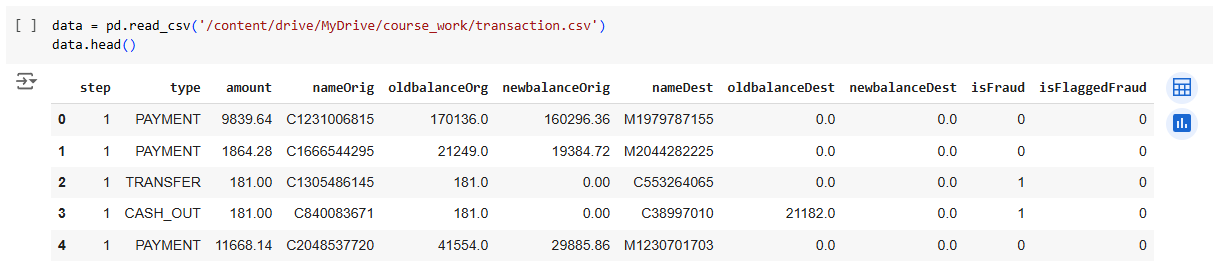
|  |  |
| --- | --- |
| **Name of the variable** | **Description** |
| step | maps a unit of time in the real world. In this case 1 step is 1 hour of time. Total steps 744 (30 days simulation). |
| type | CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER. |
| amount | amount of transaction  in local currency. |
| nameOrig | customer who started the transaction. |
| oldbalanceOrg | initial balance before the transaction. |
| newbalanceOrig | new balance after the transaction. |
| nameDest | customer who is the recipient of the transaction. |
| oldbalanceDest | initial balance recipient before the transaction. Note that there is not information for customers that start with M (Merchants). |
| newbalanceDest | new balance recipient after the transaction. Note that there is not information for customers that start with M (Merchants). |
| isFraud | This is the transactions made by the fraudulent agents inside the simulation. In this specific dataset the fraudulent behavior of the agents aims to profit by taking control or customers accounts and try to empty the funds by transferring to another account and then cashing out of the system. |
| isFlaggedFraud | The business model aims to control massive transfers from one account to another and flags illegal attempts. An illegal attempt in this dataset is an attempt to transfer more than 200.000 in a single transaction. |

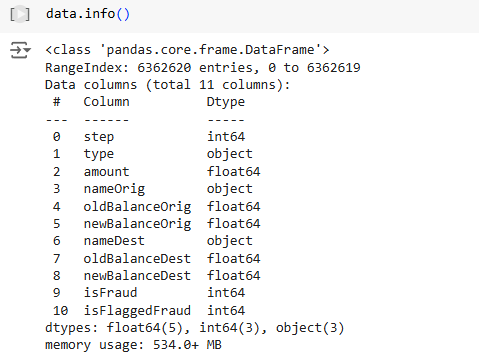
Table 1. Data definition

## Data Prossesing

The dataset contains 11 columns of information for ~6 million rows of data. The key columns available are –

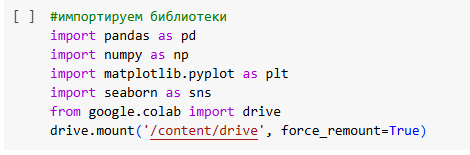
* Type of transactions
* Amount transacted
* Customer ID and Recipient ID
* Old and New balance of Customer and Recipient
* Time step of the transaction
* Whether the transaction was fraudulent or not

Picture 1. The dataframe



Picture 2. The datatype of column

Due to the need for data preparation, the machine learning model was developed in Google Colab. Google Colab is a cloud-based platform built on Jupyter Notebook that does not require installation. It offers free access to computational resources, including GPUs and TPUs, making it ideal for tasks such as machine learning, data processing, and educational projects.



Picture 3. Import libraries

The first step was to create a folder in Google Drive named "course\_work", where the dataset file downloaded from Kaggle was uploaded. Data preprocessing was then initiated. Some column names were incorrect, so they were renamed using the .rename method for consistency and clarity.

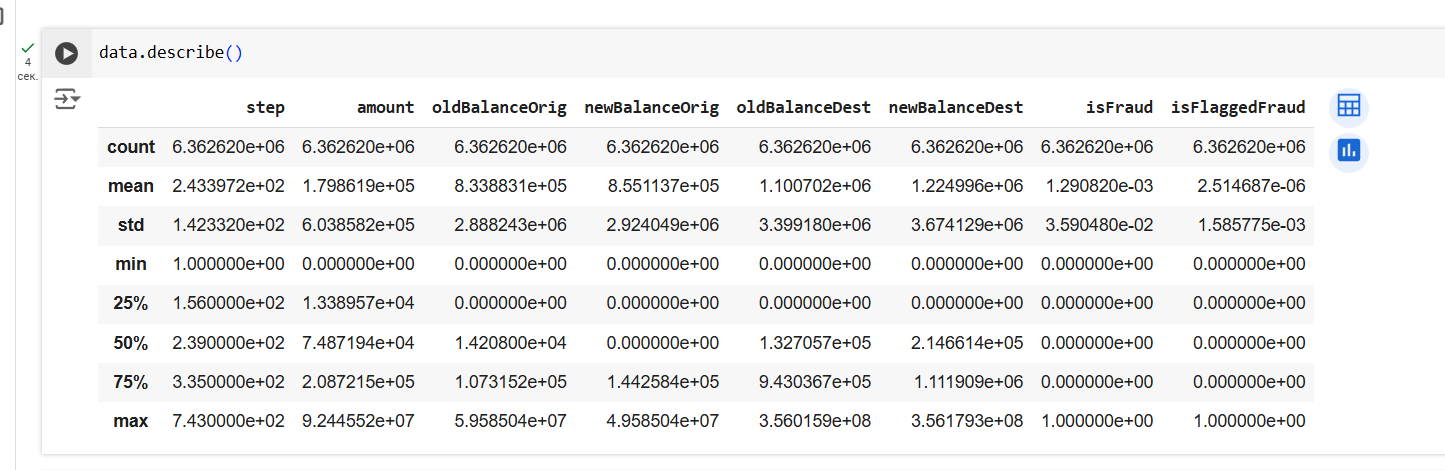
Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

Picture 4. The result of renaming.

Before analysis, it is critical to examine the summary statistics of the dataset’s numeric variables. Key metrics like the mean, standard deviation, and the range of values at different percentiles provide insights into their distribution. Below is a brief explanation of the metrics:

* Count: The total number of non-missing entries for a variable.
* Mean: The average value of a variable, calculated by summing all entries and dividing by their count.
* Standard Deviation (std): Reflects the variability of the data points around the mean. A larger value indicates more significant variation.
* Min: The smallest value observed in the dataset for the variable.
* 25% (First Quartile): The value below which 25% of the data points lie.
* 50% (Median): The midpoint value, with 50% of data points above and below it.
* 75% (Third Quartile): The value below which 75% of the data points lie.
* Max: The largest value observed in the dataset.

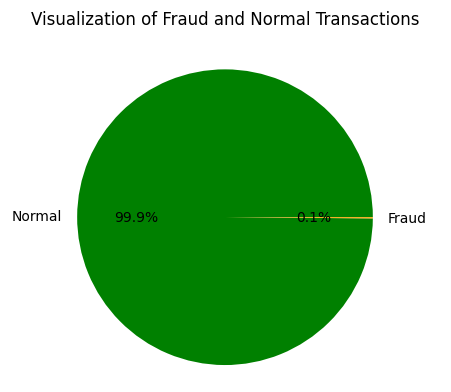
Picture 5. Describe the data.

The following conclusions follow from this table: there are more than 6 million data in this dataset.

In this exploratory analysis, we assess the class imbalance in the dataset. The class imbalance is defined as a percentage of the total number of transactions presented in the isFraud column.

The result:

* Total number of Transactions are 6362620
* Number of Normal Transactions are 6354407
* Number of fraudulent transactions are 8213
* Percentage of Fraud Transactions is 0.13
* Percentage of Normal Transactions is 99.87



Picture 6. Visualization of Fraud and Normal Transaction.

The dataset shows a significant class imbalance, with fraudulent transactions accounting for only 0.13% of the total (8,213 instances). This imbalance is critical as it implies that non-fraudulent transactions dominate the data. If a machine learning model is trained on such skewed data, it could disproportionately focus on normal transactions, potentially leading to biased results. Addressing this imbalance is essential to build a robust model capable of accurately detecting fraud.

Изображение выглядит как диаграмма, текст, снимок экрана, круг

Автоматически созданное описание

Picture 7. Transaction Type

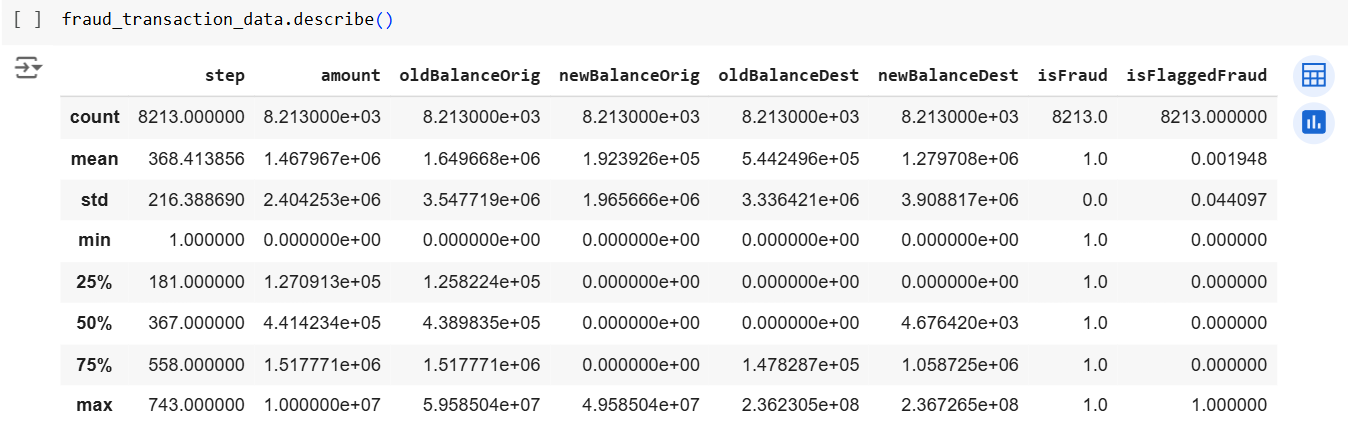
The most frequent transaction types in the dataset are as follows:

* CASH-OUT: 35.2%
* PAYMENT: 33.8%

Two specific flags in the dataset are of particular interest:

1. isFraud: This column indicates actual fraudulent transactions in the dataset.
2. isFlaggedFraud: This column shows transactions flagged by the system as potentially fraudulent due to triggered thresholds.

To analyze these in detail, the data is split into normal and fraudulent transactions, followed by detailed exploration of each type.



Picture 8. Describe the fraud transaction type.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Picture 9. Describe transaction data.

The original dataset was reduced from over 6 million transactions to approximately 2.8 million transactions for easier analysis.  
Key checks performed on the dataset:

1. Positive Transaction Amounts
   * Transactions with a negative amount: 0
   * Transactions with an amount of 0: 16

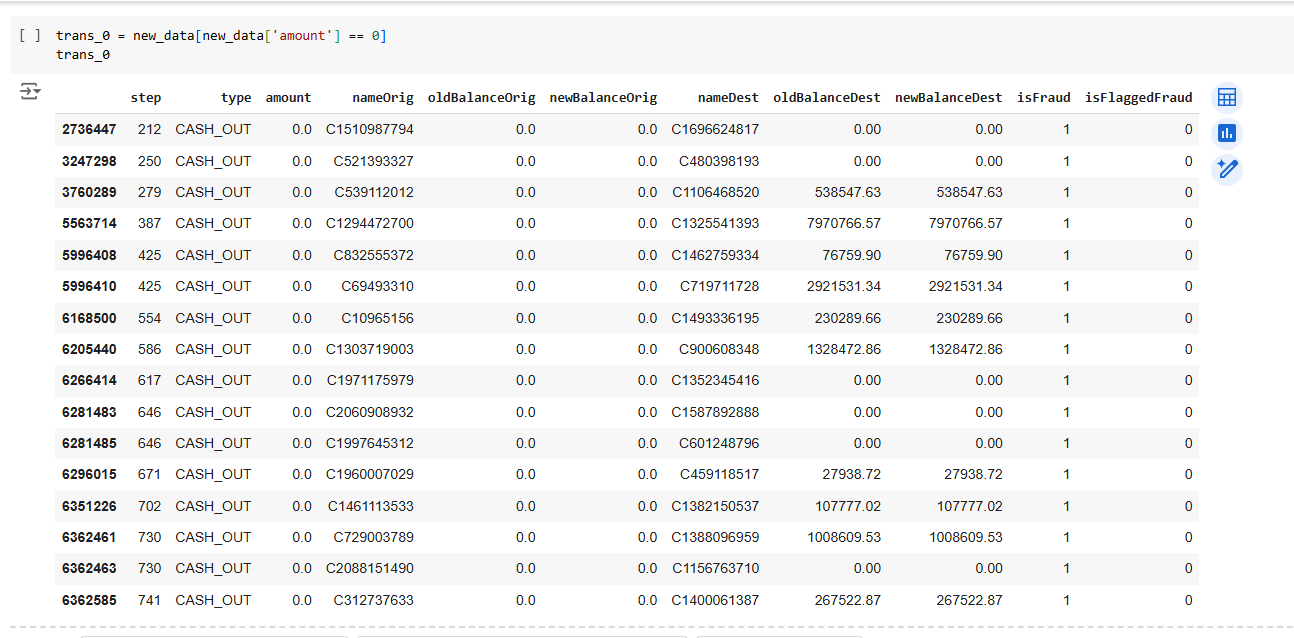
All transactions with an amount of 0 were fraudulent. Based on this observation, it can be inferred that transactions with a 0 amount are likely fraudulent.

1. Balances Check
   * Nearly half of the transactions have the originator’s initial balance (oldBalanceOrig) recorded as 0.
   * Less than 1% of cases have the recipient’s final balance (newBalanceDest) recorded as 0.

Ideally:

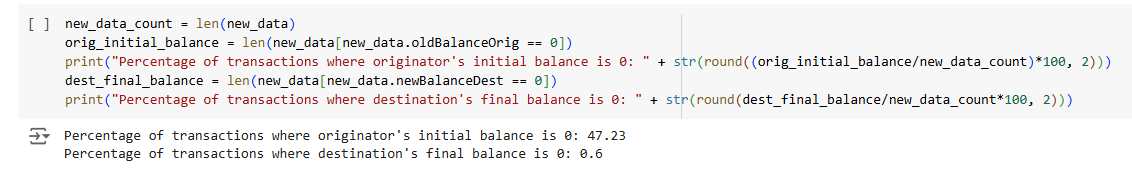
* + The recipient’s final balance should equal their initial balance plus the transaction amount.
  + The originator’s final balance should equal their initial balance minus the transaction amount.

These logical checks helped identify discrepancies and potential anomalies in the data.

Picture 10. Number of transactions where the transaction amount is 0.

There are only a few cases in which transacted amount is 0. We observe by exploring the data of these transactios that they are all fradulent transactions. So, we can assume that if the transaction amount is 0, the transaction is fraudulent.

In this section, let's check if there are any ambiguties in the originator's balance or recipient’s balance.

Picture 11. Precentage of transactions

Therefore, in almost half of the transactions, the originator's initial balance was recored as 0. However, in less than 1% of cases, the recipient's final balance was recored as 0.

Ideally, the recipient's final balance should be equal to the reipient's initial balance plus the transaction amount. Similarly, the originator's final balance s hould be equal to originator's initial balance minus the transaction amount.

So, let's check these conditions to see whether the old balance and new balance varialbes are captured accurately for both originator and recipient.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описаниеPicture 12. Final balance

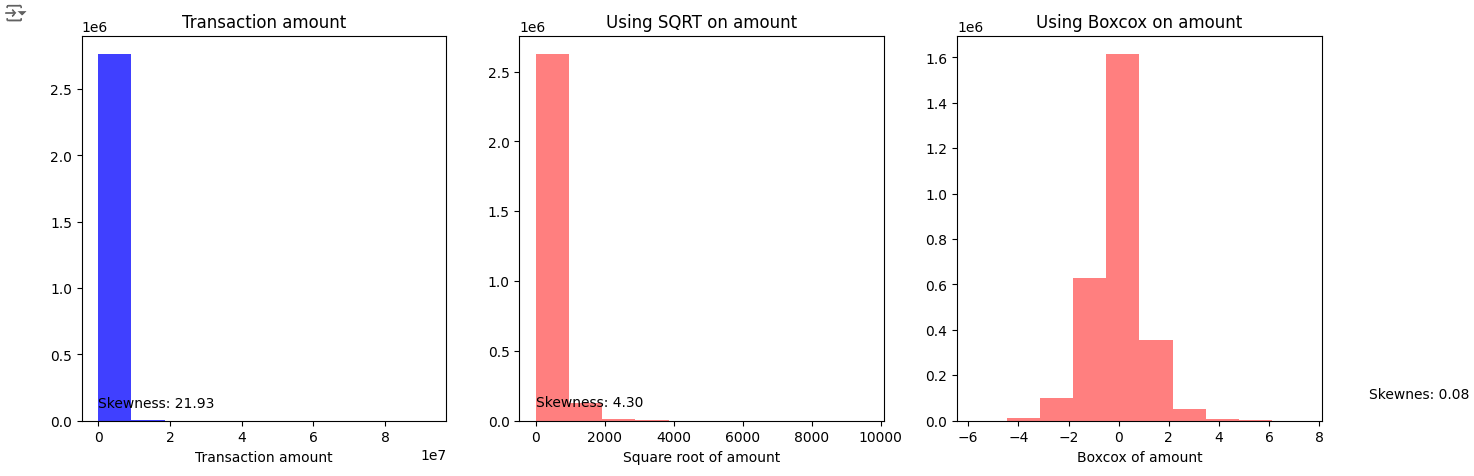
The transaction amount (amount) was highly skewed, which could negatively affect model performance. To address this, two transformations were applied:

1. Square Root Transformation: Reduces skewness by dampening larger values.
2. Box-Cox Transformation: A more sophisticated method to normalize data.

Three histograms illustrate the impact of these transformations:

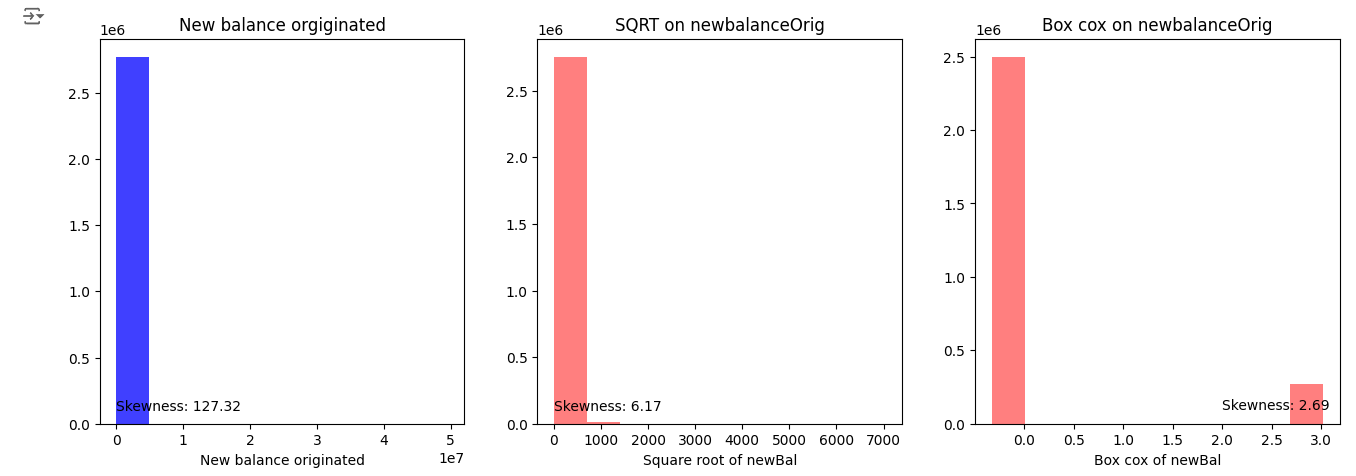
* The original data distribution is shown in blue, with a high positive skew.
* Transformed distributions are shown in red, demonstrating reduced skewness.
* Skewness values are annotated directly on the graphs to quantify the improvements.

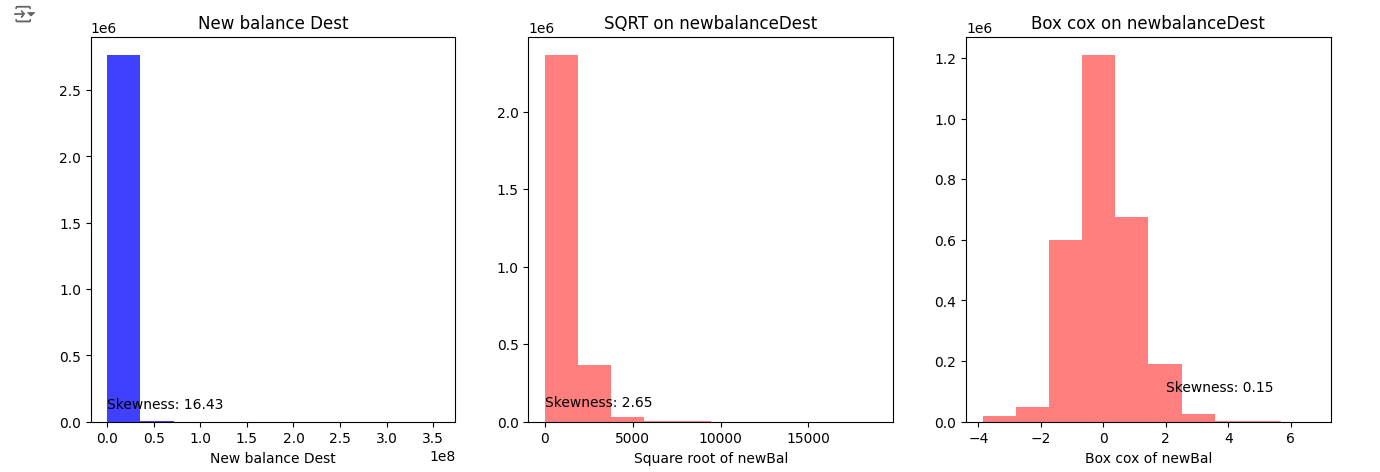
By reducing skewness, the transformed data becomes more suitable for machine learning models, which often assume normality in features.

Picture 13. Amount skewness

Изображение выглядит как текст, снимок экрана, диаграмма, График

Автоматически созданное описаниеPicture 14. Old balance skewness

Picture 15. New balance Orig skewness

Picture 16. New balance Dest

In analyzing the dataset, a major challenge arose: the significant class imbalance between normal and fraudulent transactions. Fraudulent transactions accounted for a very small percentage of the total data, making it crucial to address this imbalance to ensure the machine learning model could accurately detect fraudulent behavior. Ignoring this imbalance could lead to a model heavily biased toward the majority class, resulting in missed fraudulent transactions.

To tackle this problem, three common techniques for balancing datasets were considered:

1. Undersampling: This approach reduces the size of the majority class by randomly removing samples to balance it with the minority class. While effective in creating balance, it risks losing valuable information from the majority class.
2. Oversampling: In this method, the minority class is increased by duplicating its samples, thereby improving its representation in the dataset. Although it avoids data loss, oversampling can introduce redundancy and overfitting.
3. SMOTE (Synthetic Minority Oversampling Technique): A more advanced method that generates synthetic samples for the minority class rather than duplicating existing ones. This helps create diversity in the minority class, making it a robust choice for balancing data.

Given the computational demands of SMOTE and the simplicity of undersampling, the latter was chosen. The undersampled training dataset included all fraudulent transactions and an equal number of randomly selected normal transactions, ensuring a balanced dataset without overburdening computational resources.

The dataset was divided into training and testing subsets to train the model and evaluate its performance effectively:

* Training Data: Consisted of 80% of the original dataset, used to train the model.
* Testing Data: Comprised the remaining 20%, reserved for evaluating the model’s predictions.

The undersampled dataset underwent a similar split to maintain consistency, ensuring an equal number of fraudulent and normal transactions for training purposes.

A logistic regression model was selected for its simplicity, computational efficiency, and ability to produce interpretable results. Logistic regression is a machine learning algorithm used for binary classification. It predicts the probability of a categorical outcome (e.g., yes/no, survive/didn't survive). For example, predicting if a Titanic passenger survived based on features like age, gender, etc. The math behind logistic regression is Sigmoid or Logistic Function:

Logistic regression uses a sigmoid function to model probabilities between 0 and 1 and shaped like the letter “S”. The relationship between the original Bernoulli parameter µ and the natural parameter θ is known as the sigmoid or logistic function. Observe that µ ∈ (0, 1) but θ ∈ R, and therefore the sigmoid function squeezes a real value into the range (0, 1). This property is useful in machine learning, for example it is used not only in logistic regression, as well as a nonlinear activation functions in neural networks.

This model was trained on the balanced training dataset derived from the undersampling process.

The model's performance was evaluated using several key metrics:

1. Accuracy: Measures the overall percentage of correct predictions, including both fraudulent and normal transactions.
2. Precision: Focuses on the fraction of correctly identified fraudulent transactions out of all transactions predicted to be fraudulent.
3. Recall: Captures the proportion of actual fraudulent transactions correctly identified by the model.

In the context of fraud detection, recall is the most critical metric. Missing a fraudulent transaction (a false negative) could have severe consequences, such as financial losses or security breaches. Conversely, misclassifying a normal transaction as fraudulent (a false positive) is less problematic, as it may simply require further verification.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описаниеPicture 17. Train\_test\_split

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описаниеPicture 18. Logistic Regression

* Confusion Matrix: Provides a detailed breakdown of true positives, true negatives, false positives, and false negatives.
* Classification Report: Summarizes precision, recall, and F1-score for each class.
* ROC Curve: Visualizes the trade-off between the true positive rate and the false positive rate, with the area under the curve (AUC) measuring overall model performance.

The results demonstrate that the model achieves high recall, which aligns with the objective of capturing the majority of fraudulent transactions.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Picture 19. Seaborn Confusion Matrix with Labels

The plot above represents the Receiver Operating Characteristic (ROC) curve for a logistic regression model used to detect fraudulent transactions. This curve provides a graphical evaluation of the model's ability to distinguish between classes—in this case, fraudulent and non-fraudulent transactions.

Key Points:

1. True Positive Rate (TPR) (y-axis): Also known as recall or sensitivity, it measures the proportion of actual fraud cases correctly identified by the model.
2. False Positive Rate (FPR) (x-axis): This measures the proportion of normal transactions incorrectly classified as fraudulent.

The ROC curve shows the trade-off between the true positive rate and the false positive rate at various classification thresholds. The red dashed line represents a baseline model with no predictive power (random guessing). Any model above this line performs better than random classification.

Logistic regression is a statistical and machine learning algorithm used for binary classification tasks. It models the probability of a target variable yyy belonging to a specific class (e.g., fraudulent vs. legitimate transactions) based on one or more input features XXX. The foundation of logistic regression lies in the logit function and the sigmoid function, which maps real-valued inputs to probabilities in the range [0,1][0, 1][0,1].

Insights from the Plot:

* The ROC curve for the logistic regression model is significantly closer to the top-left corner of the plot, indicating strong performance.
* The Area Under the Curve (AUC) score, which quantifies the overall performance of the model, is 0.94. This score indicates a high level of accuracy, as a perfect model would have an AUC of 1.0, while a random model would score 0.5.

Изображение выглядит как текст, снимок экрана, линия, График

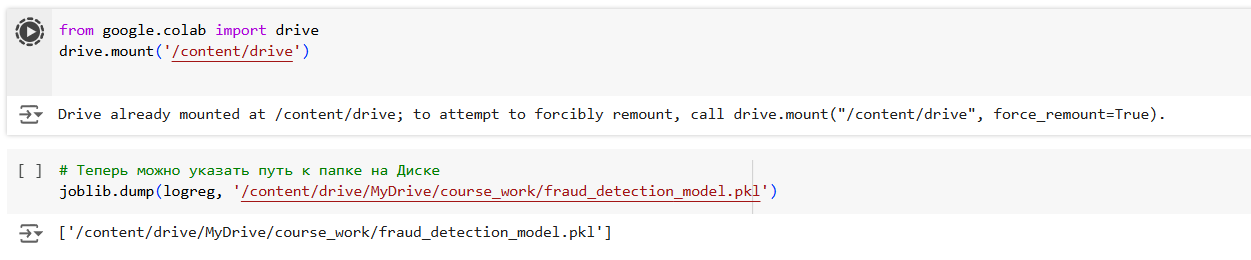
Автоматически созданное описание

Picture 20. Receiver operating characteristic

# **Client application creation**

## Database implementation

After the machine learning model was trained, I saved it as "fraud\_detection\_model.pkl" on Google Drive. From there, I downloaded it to my computer for integration into the code written in the PyCharm development environment. This step enabled the use of the trained model in the banking application to detect fraudulent transactions.

Picture 21. Saving the model of machine learning

The first focus is the creation of the database for the banking application. A database is a structured collection of data that is stored and accessed electronically. It is crucial for managing the data associated with a banking application, including customer accounts, transactions, and employee details. The chosen development environment for this task was SQL Server Management Studio 19 (SSMS), a comprehensive tool for managing SQL Server databases.

SQL Server Management Studio (SSMS) is an integrated environment that provides tools to configure, manage, and administer all components of SQL Server. It is a powerful tool for database development and administration, facilitating easy management of database objects, executing queries, and analyzing data.

Key Tables and Attributes

In this banking application, the system will manage customer, accounts, transactions, and employee data. Below is a detailed description of the key entities and their attributes:

Customer table

|  |  |
| --- | --- |
| customer\_id | int primary key identity(1,1) |
| last\_name | nvarchar(100) not null |
| first\_name | nvarchar(100) not null |
| phone | nvarchar(15) unique check(phone like '+7%' and len(phone) = 12) |
| date\_of\_birth | date |
| gender | nvarchar(20) |
| [password] | nvarchar(256) |
| registration\_date | datetime default getdate() |

Table 2. Customer table

The customer table represents the clients using the bank’s services. It stores essential personal details and contact information, such as their name, phone number, date of birth, and gender. This table also includes a password for secure access to the bank's services.

Account table

|  |  |
| --- | --- |
| account\_id | int identity(1,1) primary key |
| account\_number | nvarchar(20) unique not null check(account\_number like '4400%' and len(account\_number) = 12) |
| customer\_id | int references customer(customer\_id) |
| balance | decimal(18,2) default 0 |
| currency | nvarchar(10) default 'USD' |
| open\_date | datetime default getdate() |
| is\_active | bit default 1 |

Table 3. Account table

The accounts table represents individual bank accounts associated with customers. It tracks the balance, currency, and type of account, as well as ensuring the relationship between accounts and customers through the customer\_id.

Transactions table

|  |  |
| --- | --- |
| transaction\_id | INT IDENTITY(1, 1) PRIMARY KEY |
| account\_from | NVARCHAR(20) NOT NULL CHECK(account\_from LIKE '4400%' AND LEN(account\_from) = 12) REFERENCES accounts(account\_number) |
| account\_to | NVARCHAR(20) NOT NULL CHECK(account\_to LIKE '4400%' AND LEN(account\_to) = 12) REFERENCES accounts(account\_number) |
| transaction\_type | NVARCHAR(50) |
| amount | DECIMAL(10, 2) |
| transaction\_date | DATETIME DEFAULT GETDATE() |
| description | NVARCHAR(255) |
| status | NVARCHAR(20) DEFAULT 'pending' |
| oldBalanceOrig | DECIMAL(10, 2) |
| newBalanceOrig | DECIMAL(10, 2) |
| oldBalanceDest | DECIMAL(10, 2) |
| newBalanceDest | DECIMAL(10, 2) |
| isFlaggedFraud | BIT DEFAULT 0 |
| employee\_id | INT NULL REFERENCES employees(employee\_id) |

Table 4. Transactions table

The transactions table records every movement of money within the bank system. It tracks transfers between accounts, including the amount, type, employee\_id and status of the transaction. This table also stores information required for detecting potential fraudulent activities, such as balance changes and the use of the isFlaggedFraud flag.

Employees table

|  |  |
| --- | --- |
| employee\_id | int identity(1,1) primary key |
| firstname | nvarchar(100) not null |
| lastname | nvarchar(100) not null |
| phone | nvarchar(15) unique check(phone like '+7%' and len(phone) = 12) |
| [password] | nvarchar(256) not null |
| position | nvarchar(50) |
| hiredate | datetime default getdate() |
| salary | int |

Table 5. Employees table

The employees table stores information about the bank’s employees. It includes details such as their name, contact information, role within the bank, hire date, and salary.

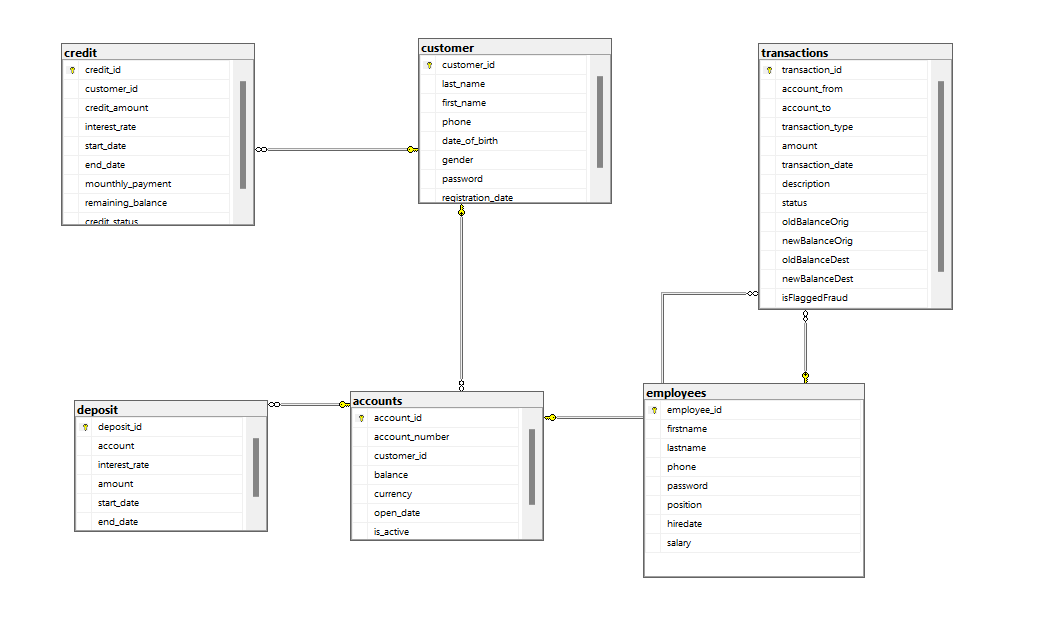
Credit table

|  |  |
| --- | --- |
| credit\_id | int identity(1,1) primary key |
| customer\_id | int references customer(customer\_id) |
| employee\_id | int references employees(employee\_id), |
| credit\_amount | decimal(10, 2) |
| interest\_rate | decimal(5, 2) |
| [start\_date] | date |
| end\_date | date |
| mounthly\_payment | decimal(10, 2) |
| remaining\_balance | decimal(10,2) |
| credit\_status | varchar(20) check (credit\_status in ('active', 'closed', 'defaulted')) |

Table 21. Credit table

The credit table implements the management of customer loans, including data on the loan amount, interest rate, maturity and status. This allows you to track active and closed loans, as well as monitor the remaining amount of debt.

Once the database schema has been designed, the BankAppDB database is created in SQL Server Management Studio (SSMS). The aforementioned tables are then created within the database, ensuring proper relationships between customers, accounts, transactions, credit and employees (picture 21).



Picture 21. Database diagrams

## Client application creation

Before developing an application using a programming language, it is essential to first create a flowchart and UML diagrams. These tools provide a structured approach to understanding the application’s functionality and design before implementation. Below are the definitions of a flowchart and UML, along with additional context regarding their significance.

A flowchart is a graphical representation of a process or workflow using standardized symbols, such as rectangles, diamonds, and arrows. It illustrates the sequence of operations or steps required to accomplish a task, making it easier to identify inefficiencies, errors, or redundancies in a process. Flowcharts are especially valuable in programming for outlining the logic of algorithms and ensuring that each step is clearly defined before coding begins.

Изображение выглядит как диаграмма, План, Технический чертеж, текст

Автоматически созданное описание

Picture 22. Flowchart

UML is a standardized modeling language used to visualize, specify, construct, and document the components of a software system. UML diagrams, such as use case diagrams, class diagrams, and sequence diagrams, provide a blueprint of the system's architecture, interactions, and behaviors. They help developers, analysts, and stakeholders communicate effectively and align their understanding of the project requirements.

Picture 23. UML

The application was developed using PyCharm, a widely-used integrated development environment (IDE) designed for Python programming. PyCharm is a powerful IDE that offers features such as code completion, debugging tools, project navigation, and integrated version control. It is an ideal environment for Python developers due to its robust support for libraries, frameworks, and productivity-enhancing tools.



Picture 24. Libraries

The following libraries were employed in the development of the application, each serving a specific purpose:

1. customtkinter and ctk (Alias for CustomTkinter):

* CustomTkinter is an extension of the Tkinter library, providing enhanced widgets and modern GUI design elements for creating visually appealing and user-friendly applications.
* It simplifies the process of styling and configuring GUI components while maintaining compatibility with Tkinter.

1. pyodbc:

* A Python module that enables interaction with databases using Open Database Connectivity (ODBC). It allows the application to execute SQL queries, retrieve data, and update records in relational databases.

1. sys:

* This module provides access to system-specific parameters and functions. It is essential for interacting with the Python runtime environment, handling command-line arguments, and managing system-level tasks like exiting the program.

1. tkinter.messagebox:

* A component of the Tkinter library used to display dialog boxes, such as error messages, warnings, or informational pop-ups. It enhances user interaction by providing clear and immediate feedback.

1. requests:

* A widely-used library for sending HTTP requests and handling responses in Python. It simplifies interaction with APIs by managing sessions, cookies, and data encoding.

1. random:

* This library generates pseudo-random numbers for various purposes, such as creating random IDs, passwords, or simulating events.

1. tkinter.simpledialog:

* A submodule of Tkinter that provides dialog boxes for simple user input. It is useful for obtaining values like names, numbers, or other data directly from users.

1. datetime:

* A built-in Python module for working with date and time objects. It is critical for logging events, scheduling tasks, or calculating durations.

1. decimal.Decimal:

* A module for performing high-precision arithmetic, especially useful for financial or scientific applications where accuracy is paramount.

1. pandas (pd):

* A versatile data manipulation library that provides powerful tools for working with structured data, such as DataFrames and Series. It is widely used in data analysis, cleaning, and preparation.

1. joblib:

* A library for serializing Python objects, particularly models, for storage or reuse. It is often employed in machine learning workflows for saving trained models.

1. scipy.stats.boxcox:

* A function in the SciPy library that applies the Box-Cox transformation to data, making it more normally distributed. This is useful in statistical modeling and machine learning preprocessing.

1. numpy (np):

* A foundational library for numerical computing in Python. It provides support for multidimensional arrays, mathematical operations, and linear algebra.

1. urllib.parse.quote\_plus:

* A utility for encoding query parameters in URLs. It ensures that special characters in strings are safely represented for web requests.

1. threading:

* A Python module for creating and managing threads, enabling concurrent execution of tasks. It is essential for improving application performance and responsiveness.

1. sqlalchemy:

* A powerful library for working with databases in Python. It provides an Object-Relational Mapping (ORM) framework and supports raw SQL queries through its create\_engine and text modules.

By leveraging these libraries, the application achieves functionality, efficiency, and user-friendliness while adhering to best practices in software development.

The presented code implements a banking application in Python using a graphical user interface (GUI) provided by the CustomTkinter library. The application connects to an SQL Server database to manage user authentication, account registration, balance management, and transactions. Below is a detailed explanation of its components and functionality.

The script uses an ODBC driver to establish a secure connection with the SQL Server database. The connection string includes parameters such as server name, database name, and security protocols:

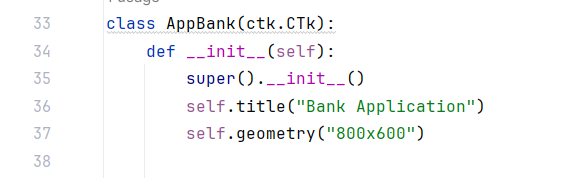
Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Picture 25. Database connection

* server and database: These define the target SQL Server and database name respectively.
* ENCRYPT=yes: Ensures that the connection is encrypted.
* TrustServerCertificate=yes: Trusts the server's self-signed certificate without further verification.
* Trusted\_Connection=yes: Indicates Windows Authentication is used.

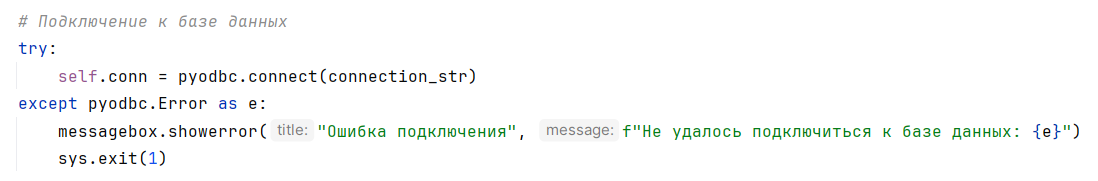
The application class AppBank inherits from ctk.CTk (CustomTkinter's main window object) and initializes the GUI:



Picture 26. Class AppBank

* self.title: Sets the application window title.
* self.geometry: Specifies the initial window dimensions.
* super().\_\_init\_\_(): Calls the parent class constructor for proper initialization.

The \_\_init\_\_ method establishes a connection to the database and handles errors gracefully:

Picture 27. \_\_init\_\_

The application uses multiple frames for modular GUI design. Key frames include:

1. Login Frame (picture 28 and picture 29): Displays user login fields and controls, such as input fields for phone number and password, along with buttons to log in or navigate to the registration frame. It also includes options to change the application's appearance mode.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Picture 28. Login Frame(light ver.)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Picture 29. Login Frame(dark ver.)

1. Register Frame (picture 30): Allows new users to register by filling out details such as their first name, last name, phone number, date of birth, gender, and password. Upon successful registration, a unique account number is generated, and the user is redirected to the login frame.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Picture 30. Register Frame

1. My Bank Frame (picture 31): Provides account details, including the user's current balance and active account information. It also offers various banking options, such as initiating transactions, deposits, credits, and accessing stock information. Additionally, it displays real-time exchange rates for selected currencies.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Picture 31. My Bank Frame

1. Transaction Frame (picture 32): Enables users to perform fund transfers. It includes input fields for the recipient's account number, transfer amount, and an optional description. The frame also performs validation checks and records the transaction details.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Picture 32. Transaction Frame

1. Employee Frame (picture 33): Designed specifically for employees, this frame allows access to employee-related functionalities such as managing customer accounts, handling approvals, and overseeing banking operations. Employee-specific data is loaded upon login.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Picture 33. Employee Frame

1. Deposit Frame (picture 34): Facilitates deposits into the user's account. Users can specify the deposit amount and, upon confirmation, the balance is updated accordingly. This frame ensures that deposits are validated and securely processed.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Picture 34. Deposit Frame

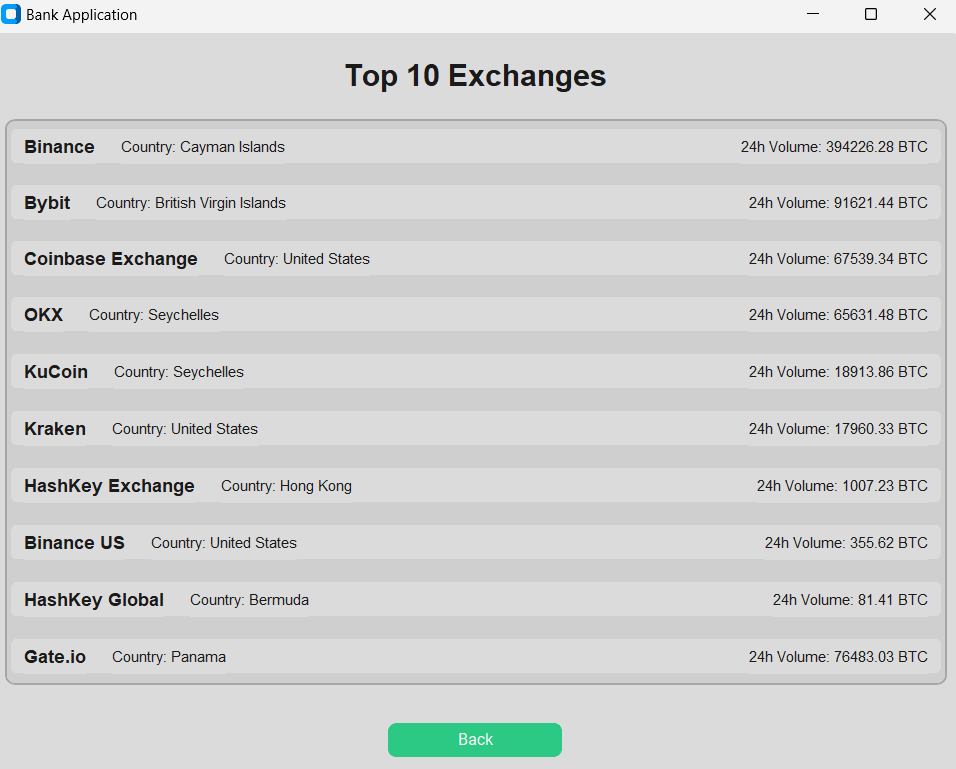
1. Credit Frame (picture 35): Provides options for users to manage credit-related services, such as applying for loans, viewing credit limits, and monitoring repayment schedules.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Picture 35. Credit Frame

1. Stock Frame (Picture 36): top 10 exchanges.



Picture 36. Stock Frame

Each frame is carefully designed to encapsulate specific functionalities, ensuring modularity, clarity, and ease of navigation within the banking application.

## Machine learning implementation

One of the core features of the application is its Fraud Detection System, which leverages a machine learning model trained to identify fraudulent transactions. Below, we outline the system's design and functionality.

This system enables secure money transfers between users while ensuring that any potential fraudulent transactions are flagged and handled appropriately. The process is structured to involve both the user and an employee, each having a specific role in the transaction lifecycle.

The process begins when a user initiates a money transfer to another user. The system collects necessary details, such as the transaction amount and recipient account information. Initially, the transaction enters a state marked as pending, meaning that it is queued for further processing. During this stage, the system does not yet move the money between accounts, as the transaction must undergo fraud detection to ensure its legitimacy. The transaction's status is stored in the database as 'pending' to signify that it has not been completed (picture 37 and picture 38).

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Picture 37. Successfully transaction

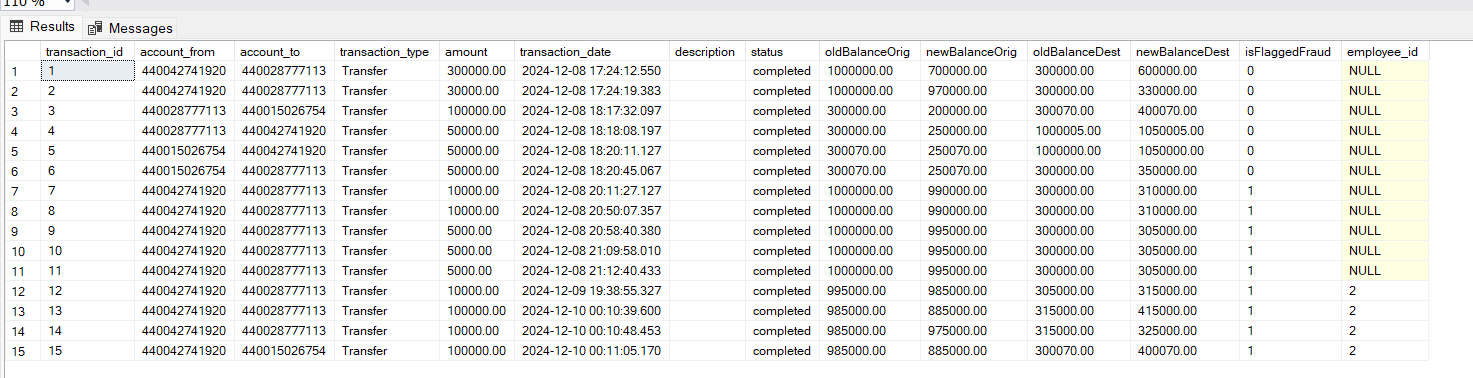
Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Picture 38. select \* from transactions

After the transaction is created and marked as pending, the next step involves an employee triggering the fraud detection process. When the fraud detection process is started, the system retrieves the relevant transaction data and uses a machine learning model to predict the likelihood of the transaction being fraudulent. The model evaluates various features of the transaction, such as the amounts involved and the account balances before and after the transaction.

If the model identifies the transaction as potentially fraudulent, the employee is notified, and the transaction can be flagged as fraudulent. If, however, the model indicates that the transaction is highly likely to be legitimate (with a fraud probability of less than 90%), the transaction is marked as completed. At this point, the status of the transaction is updated from 'pending' to 'completed' (picture 39).



Picture 39. select \* from transaction after fraud detection

Key changes occur after the fraud detection process is completed:

* Transaction Status Update: The transaction's status is changed to 'completed', signifying that the transaction has been processed and validated as legitimate.
* Balance Adjustment: The account balances for both the sender and the recipient are updated in the system. The sender's account balance is adjusted according to the original transaction amount, and the recipient's balance is updated to reflect the transfer. These changes are made by updating the accounts table in the database, where the balances for both accounts are set to the new values calculated from the transaction.

This dual-step process ensures that money is only transferred after the transaction has been validated and cleared of any fraud risks, ensuring the security and integrity of the financial system.

# **Conclusion**

The detection of fraudulent activities in financial transactions is a critical area of focus in modern banking and finance. This project explored the application of machine learning algorithms to identify anomalous transactions and distinguish between legitimate and fraudulent activities. By analyzing transactional data and implementing predictive models, we have demonstrated the potential of machine learning to enhance security and minimize financial losses for institutions and customers alike.

The integration of machine learning into fraud detection systems offers several key advantages, including the ability to process large volumes of data efficiently and uncover subtle patterns that traditional rule-based systems might overlook. Our study highlights the importance of selecting appropriate features and algorithms, as well as the necessity of ensuring data quality and balancing datasets to avoid biases that could hinder model performance.

Despite its advantages, machine learning in fraud detection is not without challenges. Fraudulent behaviors are constantly evolving, requiring continuous model updates and retraining to maintain accuracy. Furthermore, issues related to data privacy, transparency, and explainability must be addressed to ensure the ethical application of these technologies. Collaboration between data scientists, domain experts, and regulatory authorities is crucial for developing robust, trustworthy systems.

In conclusion, machine learning represents a transformative approach to combating financial fraud, offering significant improvements over traditional methods. However, its effectiveness depends on a combination of technological innovation, ethical considerations, and ongoing vigilance. By leveraging these tools responsibly, financial institutions can build a safer and more resilient financial ecosystem.

# **Reference**

* 1. Aggarwal, C. C. (2015). *Outlier Analysis*. Springer.
  2. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
  3. Jindal, A., & Liu, B. (2007). "Identifying Comparative Sentences in Text Documents". *Proceedings of the 29th European Conference on Information Retrieval (ECIR 2007)*.
  4. Sahin, Y., & Duman, E. (2011). "Detecting Credit Card Fraud by Decision Trees and Support Vector Machines". *Proceedings of the International MultiConference of Engineers and Computer Scientists*.
  5. Ngai, E. W. T., Hu, Y., Wong, Y. H., Chen, Y., & Sun, X. (2011). "The Application of Data Mining Techniques in Financial Fraud Detection: A Classification Framework and an Academic Review of Literature". *Decision Support Systems*, 50(3), 559-569.
  6. Cortes, C., & Vapnik, V. (1995). "Support-Vector Networks". *Machine Learning*, 20(3), 273-297.
  7. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Research*, 12, 2825-2830.
  8. Goldstein, M., & Uchida, S. (2016). "A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data". *PLOS ONE*, 11(4).
  9. Financial Action Task Force (FATF). (2023). "Guidance on Risk-Based Supervision". Retrieved from [www.fatf-gafi.org](http://www.fatf-gafi.org).
  10. Zhang, Y., & Jiang, J. (2020). "Fraud Detection in Financial Transactions Using Machine Learning: A Review". *Journal of Big Data*, 7(1), 1-21.