**Egyptian E-Learning University**

Faculty of Computers & Information Technology

Credit Card Fraud Detection

**By**

|  |  |
| --- | --- |
| Name: Nada Ashraf Mohamed | ID: 21-01333 |
| Name: Angi Amgad Fawzy | ID: 21-00507 |
| Name: Naiera Farag Elamir | ID: 21-01349 |
| Name: Mennatollah Mohamed Awad | ID: 21-01080 |
| Name: Mariam Walid Ahmed | ID: 21-00997 |
| Name: Ahmed Abdelsatar Ahmed | ID: 21-00377 |
| Name: Kerolos Wael | ID: 20-01849 |

Supervised by

Dr. Shaimaa Ashmawy

Assistant

Eng. Ahmed Eissa

[Hurghada]-2025

Contents

[Abstract](#_Toc197463970)

Chapter 1: [Introduction](#_Toc197463972)

1.1 Introduction

1.2 Background and Motivation

1.3 Importance of the Problem

1.4 Problem Statemen

1.5 Objectives

1.6 Overview of the Proposed Solution

Chapter 2: [Literature Review / Related Work](#_Toc197463973)

Chapter 3:[Proposed system](#_Toc197463974)

3.1 Approach Used

3.2 System Architecture

3.3Algorithms/frameworks used

3.4 Frameworks used

Chapter 4: [Implementation](#_Toc197463975)

4.1 Technologies, tools, and programming languages used

4.2 Key components/modules of the system

4.2.1 Dataset Description

4.2.2 Data Preprocessing Steps

4.3 Challenges faced and how they were resolved

Chapter 5: [Testing & Evaluation](#_Toc197463976)

5.1 Testing strategies (unit testing, integration testing, user testing)

5.2 Performance metrics (accuracy, speed, scalability, etc.)

5.3 Comparison with existing solutions (if applicable)

Chapter 6: [Results & Discussion](#_Toc197463977)

6.1 Introduction

6.2 Summary of findings

6.3 Interpretation of results (Did the project meet its objectives?)

6.4 Limitations of the proposed solution

7: [Conclusion & Future Work](#_Toc197463978)

7.1 Summary of contributions

7.2 Possible improvements or extensions for future work

[References](#_Toc197463979)

[Appendices](#_Toc197463980)

**List of Figures**

Fig3.2 System Architecture………………………………………………………………………… Fig4.1 Use case…………………………………………………………………………………….. Fig. 1 Displaing Dataset…………………………………………………………………………… Fig1.2………………………………………………………………………………………………. Fig. 1.3 Transactions Types Distribution…………………………………………………………... Fig. 2 Data Preprocessing………………………………………………………………………….. Fig. 2.2……………………………………………………………………………………………... Fig. 3 Data Splitting………………………………………………………………………………... Fig. 4 Machine Learning Models…………………………………………………………………... Fig. 4.1……………………………………………………………………………………………... Fig. 5 Deep Learning Model………………………………………………………………………. Fig. 5.1……………………………………………………………………………………………... Fig. 6 System Architecture…………………………………………………………………………. Fig. 7 Website Front-End…………………………………………………………………………... Fig.8WebsiteBackend………………………………………………………………………………Fig. 8.1……………………………………………………………………………………………... Fig. 8.2……………………………………………………………………………………………...

**Abstract**

Credit card fraud presents a serious threat to the financial industry, resulting in billions of dollars in losses globally every year. As online transactions become more prevalent, the need for accurate and intelligent fraud detection systems has become increasingly critical. This project aims to build a robust system capable of detecting fraudulent transactions using advanced machine learning and deep learning techniques. We implemented and evaluated three models: Logistic Regression, Decision Tree Classifier, and a Dense Neural Network (DNN). The models were assessed using performance metrics such as accuracy, precision, recall, and F1-score.

To address the issue of class imbalance in the dataset, we applied the SMOTE technique and used label encoding for preprocessing. The final system was deployed using a Flask-based backend and integrated with a simple payment interface, allowing for real-time fraud prediction. Results showed that while the Decision Tree achieved the highest accuracy, the DNN offered superior precision, making it highly effective for complex fraud detection scenarios.

**Chapter 1**

Introduction

**1.1 Introduction**

Credit card fraud is the unauthorized use of another person’s credit card information to conduct financial transactions without the cardholder’s consent. As digital commerce continues to expand, the frequency and complexity of fraudulent activities have increased significantly.  
According to the *Nilson Report*, global losses due to credit card fraud reached **$32.3 billion in 2021**, and projections indicate this figure could surpass **$43 billion by 2026**. A substantial portion of these losses stem from *card-not-present* (CNP) transactions, such as those made online, where it is more difficult to authenticate the user.

**1.2 Background and Motivation**

Online payment platforms have revolutionized the way financial transactions are conducted, offering speed and convenience. However, these benefits come at the cost of increased vulnerability to cybercrime. Traditional fraud detection systems rely heavily on rule-based mechanisms, which are often inadequate in identifying novel or subtle fraud patterns. This growing concern motivates the development of intelligent systems capable of learning and adapting to detect fraudulent transactions with higher accuracy and lower false positive rates.

**1.3 Importance of the Problem**

Credit card fraud results in significant financial losses and erodes consumer trust in digital financial systems. For financial institutions and online vendors, the reputational damage and financial burden from fraud cases can be severe. Developing accurate, automated, and scalable fraud detection systems is essential to mitigate risk, ensure secure transactions, and maintain user confidence in digital payment systems.

**1.4 Problem Statement**

The primary challenge in credit card fraud detection lies in the imbalanced nature of transactional data, where fraudulent instances are extremely rare compared to legitimate ones. Moreover, fraudulent patterns are constantly evolving, rendering traditional detection methods ineffective. Thus, there is a pressing need for machine learning and deep learning approaches that can handle imbalanced data and detect complex fraud behaviors in real time.

**1.5 Objectives**

* **Main Objective:**  
  Develop and deploy an intelligent credit card fraud detection system using machine learning and deep learning techniques.
* **Specific Objectives:**
  + Preprocess and balance the dataset using appropriate statistical techniques (e.g., SMOTE).
  + Train and evaluate multiple classification models such as Logistic Regression, Decision Tree, and Deep Neural Networks.
  + Deploy the best-performing model in a web-based system for real-time fraud detection.

**1.6 Overview of the Proposed Solution**

This project presents a complete fraud detection system integrated into a web-based payment interface. It employs three classification models—Logistic Regression, Decision Tree, and Dense Neural Network.

**Chapter 2**

Literature Review/Related Work

* Many existing fraud detection systems use traditional rule-based approaches, which fail with evolving fraud tactics.
* Recent works employ ML techniques but often ignore data imbalance.
* Our project addresses imbalance with SMOTE and evaluates deep learning effectiveness.

**Chapter 3**

**Proposed System**

**3.1 Approach Used**

To address the problem of detecting fraudulent credit card transactions, we followed a machine learning–based approach. The main steps of our methodology were:

1. Data Collection and Understanding

We used a publicly available dataset containing thousands of online transaction records labeled as fraudulent or legitimate. Initial exploration helped us identify key features and the extent of class imbalance

2. Data Preprocessing

We cleaned the dataset by removing irrelevant columns, encoded categorical variables using Label Encoding, and scaled numeric features. To address severe class imbalance, we applied the SMOTE technique to generate synthetic minority class samples.

3. Model Selection and Training

We implemented and trained three models: Logistic Regression, Decision Tree Classifier (both using Scikit-learn), and a Dense Neural Network (using Keras/TensorFlow). Each model was trained on the same data and tuned using standard hyperparameters.

4. Evaluation and Comparison

All models were evaluated using accuracy, precision, recall, and F1-score. We compared performance on a test set to determine which model performs best in identifying fraudulent transactions.

5. Deployment

The best-performing model (DNN) was integrated into a Flask backend, allowing real-time predictions based on user-submitted transactions via a web form.

This structured approach ensured our system was both accurate and deployable for real-world usage.

**Fig 3.2 System Architecture:**

A diagram of a system architecture

AI-generated content may be incorrect.

- Frontend: A simple HTML/CSS payment form where users can enter transaction details such as amount, sender balance, recipient balance, and transaction type.

- Backend: A Flask-based RESTful API that receives transaction data from the frontend. It preprocesses the input data, loads the trained Dense Neural Network model, and predicts whether the transaction is fraudulent or not.

- Machine Learning Model: A DNN model trained on preprocessed and balanced transaction data. It outputs a binary classification (0 = legitimate, 1 = fraud).

- Response: The backend sends the prediction result (e.g., "Fraudulent transaction detected") back to the frontend for display.

This architecture ensures a real-time, user-friendly, and scalable solution for detecting fraudulent transactions during payment submission.

The proposed system follows a modular architecture that includes a simple frontend interface, a backend API, and a trained machine learning model for fraud detection. The major components are:

**3.3 Algorithms/frameworks used:**

**1) Logistic Regression**

Logistic regression is a classification algorithm used to find the probability of event success and event failure. It is used when the dependent variable is binary (0/1, True/False, Yes/No) in nature. It supports categorizing data into discrete classes by studying the relationship from a given set of labelled data. It learns a linear relationship from the given dataset and then introduces a non-linearity in the form of the Sigmoid function.

**Advantages of this model:**

Logistic regression is easier to implement, interpret, and very efficient to train.

It makes no assumptions about distributions of classes in feature space.

It can easily extend to multiple classes (multinomial regression) and a natural probabilistic view of class predictions.

It not only provides a measure of how appropriate a predictor (coefficient size) is, but also its direction of association (positive or negative).

It is very fast at classifying unknown records.

**Disadvantages of this model:**

If the number of observations is less than the number of features, Logistic Regression should not be used, otherwise, it may lead to overfitting.

It constructs linear boundaries

The major limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables.

It can only be used to predict discrete functions.

Hence, the dependent variable of Logistic Regression is bound to the discrete number set.

Non-linear problems can't be solved with logistic regression because it has a linear decision surface. Linearly separable data is rarely found in real-world scenarios.

**2) Decision tree**

Decision tree regression is a machine learning algorithm used for predictive modeling. It operates by recursively partitioning the dataset into subsets based on the values of input features, creating a hierarchical tree-like structure. Each internal node of the tree represents a decision based on a specific feature, leading to a subsequent split, while the leaf nodes contain the predicted numerical outcome.

**Advantages of this model:**

Interpretability: The decision rules learned by the algorithm are easy to understand and visualize, making it simple to explain to non-technical stakeholders.

Non-linear relationships: Decision trees can capture non-linear relationships between features and the target variable, making them suitable for datasets with intricate patterns.

No feature scaling required: Decision trees are not sensitive to the scale of features, there's no need for feature scaling (e.g., normalization or standardization) as required by some other algorithms.

Handles both numerical and categorical data: Decision trees can handle both numerical and categorical features without the need for one-hot encoding or other preprocessing techniques. This makes them convenient for datasets with mixed data types.

**Disadvantages of this model:**

Overfitting: Decision trees are prone to overfitting, especially when they grow too deep or when the dataset is noisy.

High variance: Decision trees have high variance, meaning small changes in the training data can result in significantly different trees.

Instability: Decision trees are sensitive to small variations in the data, which can lead to different splits and, consequently, different trees. This instability makes them less reliable compared to some other algorithms.

**3)Dense Neural Network (DNN):**

Dense Neural Network is a Deep learning technique. It is a type of artificial neural network with multiple hidden layers between the input and output layers. Each layer is made up of interconnected nodes (neurons) that apply transformations to the input data.

**Advantages of this model:**

DNNs are capable of learning complex patterns, representations and can model highly non-linear relationships and extract intricate patterns in large and complex datasets, making them effective for tasks like image recognition, natural language processing, and speech recognition.

Its biggest advantage is that it’s more accurate. When trained properly and with enough data, DNNs often outperform traditional machine learning models in tasks like image and speech recognition.

**Disadvantages of this model:**

High computational cost: Training deep learning models requires significant computational resources, including powerful GPUs and large amounts of memory. This can be costly and time-consuming.

Overfitting: Overfitting occurs when a model is trained too well on the training data and performs poorly on new, unseen data. This is a common problem in deep learning, especially with large neural networks, and can be caused by a lack of data, a complex model, or a lack of regularization.

Lack of interpretability: Deep learning models, especially those with many layers, can be complex and difficult to interpret. This can make it difficult to understand how the model is making predictions and to identify any errors or biases in the model.

**3.4 Frameworks used:**

Scikit-learn → For Logistic Regression & Decision Tree

Keras + TensorFlow → For Deep Neural Network

Pandas & NumPy → For data manipulation

Flask → For deploying the trained model in a web API

**Chapter 4**

Implementation

**Fig. 4.1 Use case**

A diagram of a credit card transaction

AI-generated content may be incorrect.

**4.1 Technologies, tools, and programming languages used:**

In this project, we used a variety of tools and technologies to implement and deploy the fraud detection system:

Programming Language: Python

Libraries/Frameworks:

scikit-learn: For building and evaluating Logistic Regression and Decision Tree models.

TensorFlow/Keras: For designing, training, and deploying the Deep Neural Network (DNN).

pandas / numpy: For data preprocessing and manipulation.

matplotlib / seaborn: For data visualization and graph plotting.

imblearn: Specifically the SMOTE technique to handle data imbalance.

Deployment Tools:

Flask: For building the backend API that hosts the trained model.

HTML/CSS/JavaScript: To build a simple web-based payment interface.

Google Colab: As the development and training environment.

**4.2 Key components/modules of the system:**

The system is composed of the following core modules:

Data Preprocessing Module:

Handles label encoding, SMOTE balancing, and splitting the data into training, validation, and test sets

Model Training Module:

Trains three different models: Logistic Regression, Decision Tree, and Dense Neural Network (DNN), using the processed dataset.

Evaluation Module:

Calculates key performance metrics such as accuracy, precision, recall, and F1-score, and generates visualizations like confusion matrices

Prediction API (Flask):

Hosts the trained DNN model and receives transaction data through a RESTful API to make real-time predictions.

Frontend Interface:

A simple HTML web page that simulates a payment form, allowing users to enter transaction data and view prediction results via the Flask backend.

History Logging Module:

Stores each transaction and its prediction result in a local JSON file for future analysis or auditing.

**4.2.1 Dataset Description:**

The dataset used is the "Online Payment Fraud Detection Dataset" from Kaggle. It includes 6,362,620 records and 11 features, such as:

1. step: represents a unit of time where 1 step equals 1 hour
2. type: type of online transaction
3. amount: the amount of the transaction
4. nameOrig: customer starting the transaction
5. oldbalanceOrg: balance before the transaction
6. newbalanceOrig: balance after the transaction
7. nameDest: recipient of the transaction
8. oldbalanceDest: initial balance of recipient before the transaction
9. newbalanceDest: the new balance of recipient after the transaction
10. isFraud: fraud transaction

Fraudulent transactions account for only 0.13% of the dataset, making it highly imbalanced.

**4.2.2 Data Preprocessing Steps:**

To prepare the dataset for model training, several preprocessing techniques were applied:

Label Encoding:

Converted the categorical type column into numerical values.

SMOTE (Synthetic Minority Oversampling Technique):

Used to balance the dataset by synthetically generating minority class (fraud) samples, resulting in equal counts of fraud and legitimate transactions.

Data Splitting:

The data was split into:

80% Training

20% Testing

The training set was further divided into training and validation sets for DNN training optimization

**4.3 Challenges faced and how they were resolved:**

Challenges Faced and How They Were Resolved

Throughout the development process, we encountered several challenges:

Imbalanced Dataset:

The number of fraudulent transactions was significantly smaller than valid ones, which made training biased.

Resolved by applying SMOTE to generate synthetic fraud samples and balance the dataset.

Overfitting in DNN:

The deep neural network initially overfit the training data.

Resolved using regularization (L2), dropout layers, early stopping, and batch normalization.

Model Deployment:

Integrating the trained DNN with a real-time web interface was technically challenging.

Resolved by using Flask to serve the model as an API and AJAX calls to communicate with the frontend.

Training Time:

Training the DNN on large data was time-consuming.

Resolved by using Google Colab’s GPU environment to accelerate training.

**Chapter 5**

Testing & Evaluation

**5.1 Testing strategies (unit testing, integration testing, user testing):**

* Unit testing: Validate individual modules
* Integration testing: Validate frontend-backend-model flow
* User testing: Simulated users submitted transactions via frontend

**5.2 Performance metrics (accuracy, speed, scalability, etc.):**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Logistic Regression | Decision Tree | DNN |
| Accuracy | 0. 947629 | 0.999379 | 0.9597 |
| Precision | 0. 947606 | 0.9993792 | 0.9833 |
| Recall | 0. 947601 | 0.9993791 | 0.9352 |
| F1 Score | 0. 9476015 | 0.9993791 | 0.9573 |

**5.3 Comparison with existing solutions (if applicable):**

Traditional systems use fixed rules.

Our ML/DL approach adapts to new fraud patterns.

Real-time prediction and front-end integration is an advantage.

**Chapter 6**

Results & Discussion

**6.1 Introduction:**

This chapter presents the experimental results of the models and discusses their implications.

**6.2 Summary of findings:**

\* All models performed well

\* Decision Tree had highest accuracy

\* DNN was best for precision and recall

**6.3 Interpretation of results (Did the project meet its objectives?):**

The project met its objectives by creating a system that accurately detects fraud and is usable in real-time environments.

**6.4 Limitations of the proposed solution:**

* Computational cost of DNN
* SMOTE may overgeneralize minority class

**Chapter 7**

Conclusion & Future Work

**7.1 Summary of contributions.**

Developed a full fraud detection pipeline using ML and DL.

Deployed system with real-time capability.

Compared multiple models under fair metrics.

**7.2 Possible improvements or extensions for future work.**

Use real-time streaming data

Explore advanced deep learning models like LSTM

Deploy on scalable cloud infrastructure

References

[1] “Online Payments Fraud Detection Dataset.” Kaggle, 17 Apr. 2022, [www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset](http://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset).

[2] “1.10. Decision Trees.” Scikit-learn, [scikit-learn.org/stable/modules/tree.html](http://scikit-learn.org/stable/modules/tree.html)

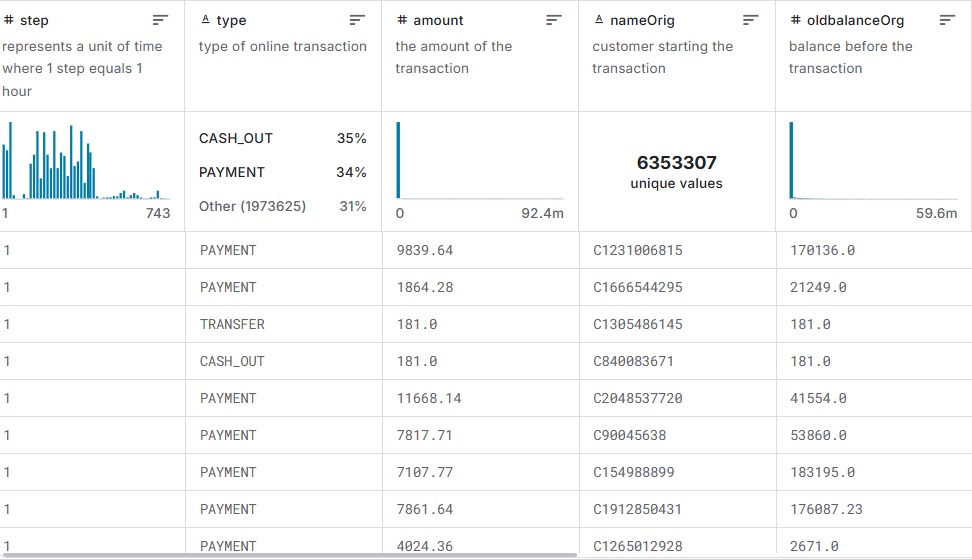
[3] Team, Keras. Keras Documentation: Dense Layer. [keras.io/api/layers/core\_layers/dense](http://keras.io/api/layers/core_layers/dense).

[4] Welcome to Flask — Flask Documentation (3.1.x). [flask.palletsprojects.com](http://flask.palletsprojects.com/).

[5] Koul, Anirudh, et al. “Practical Deep Learning for Cloud, Mobile, and Edge.” O’Reilly Online Learning, [www.oreilly.com/library/view/practical-deep-learning/9781492034858](http://www.oreilly.com/library/view/practical-deep-learning/9781492034858).

Appendices

**Fig. 1 Displaying Dataset**



**Fig. 1.2**

A screenshot of a computer

AI-generated content may be incorrect.

**Fig. 1.3**

A diagram of a transaction type distribution

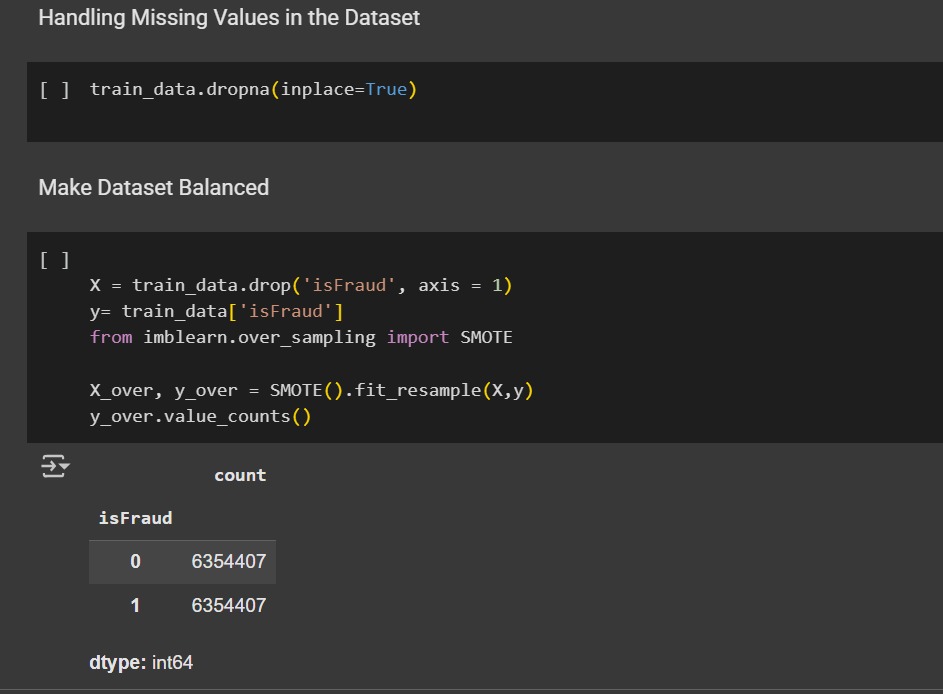
AI-generated content may be incorrect.

**Fig. 2 Data Preprocessing**

A screenshot of a computer

AI-generated content may be incorrect.

**Fig. 2.2**

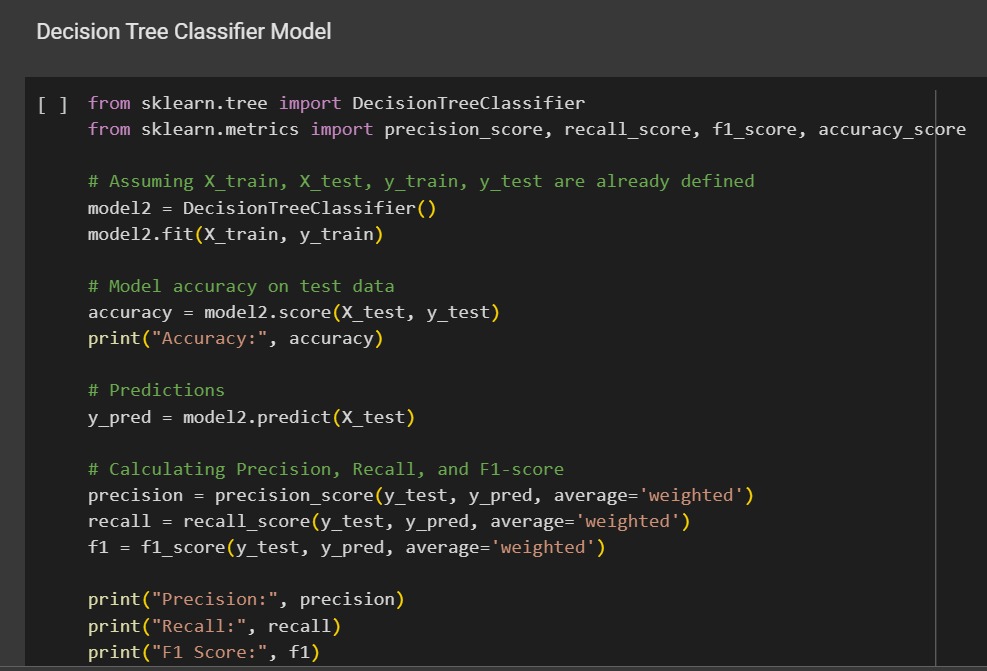


**Fig. 3 Data Splitting**

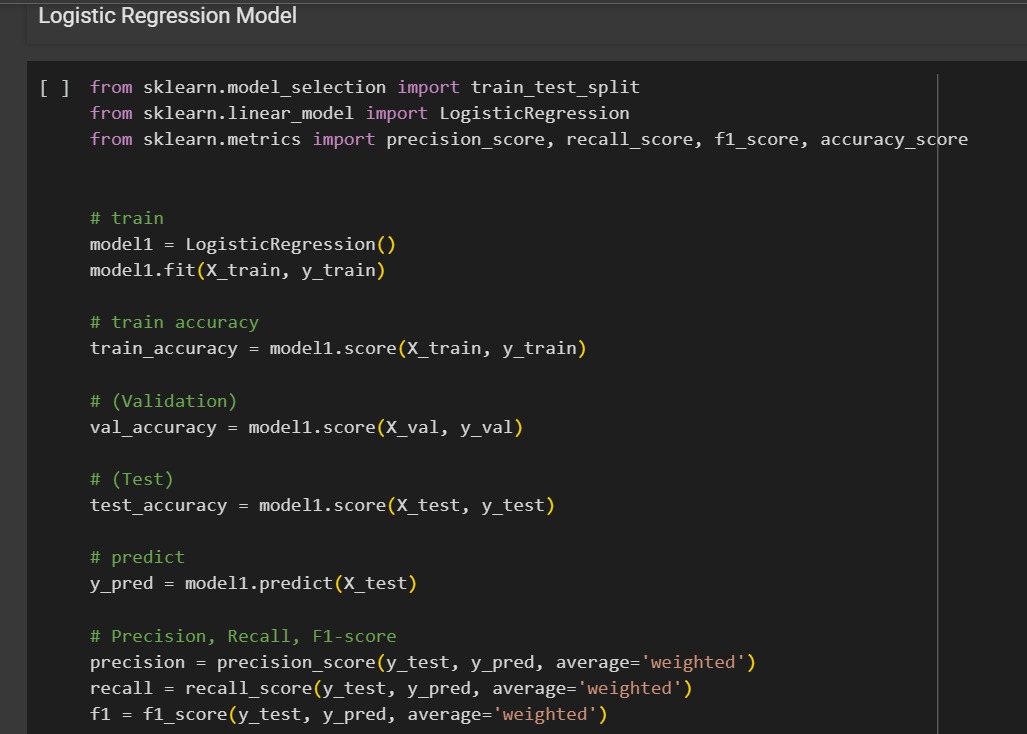
A screenshot of a computer program

AI-generated content may be incorrect.

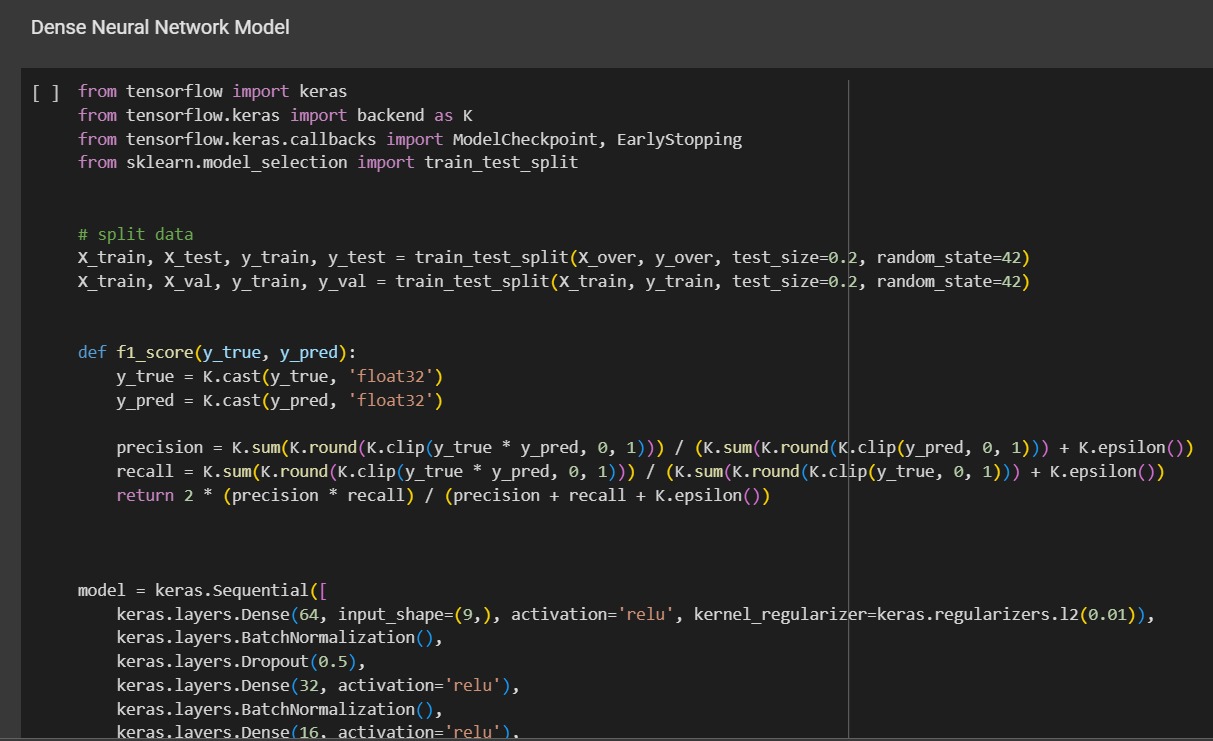
**Fig. 4 Machine Learning Models**



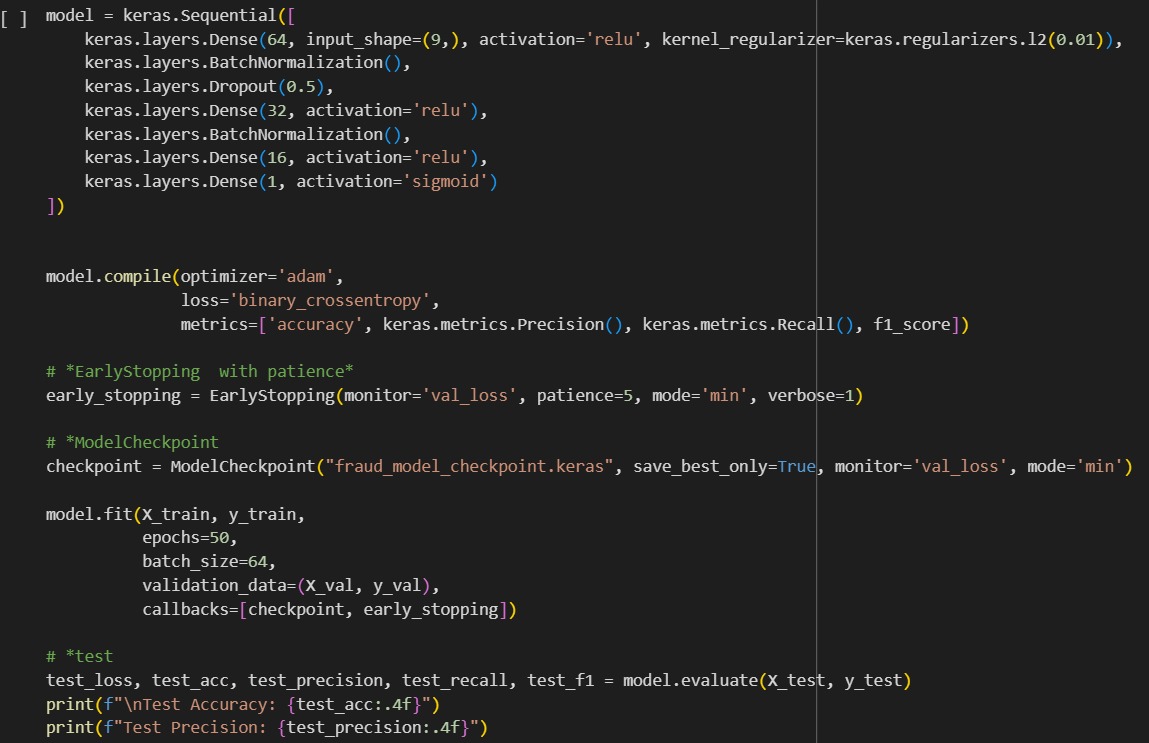
**Fig. 4.1**



**Fig. 5 Deep Learning Model**



**Fig. 5.1**

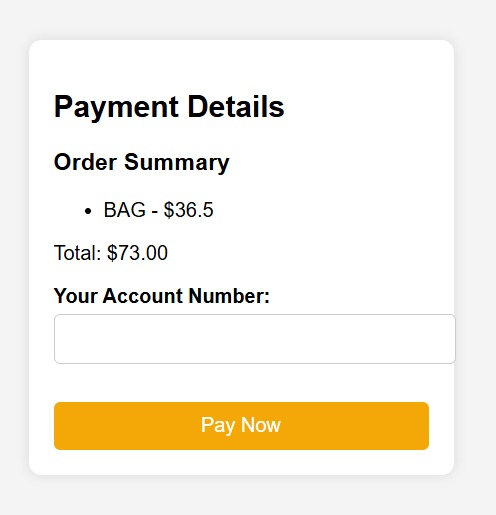


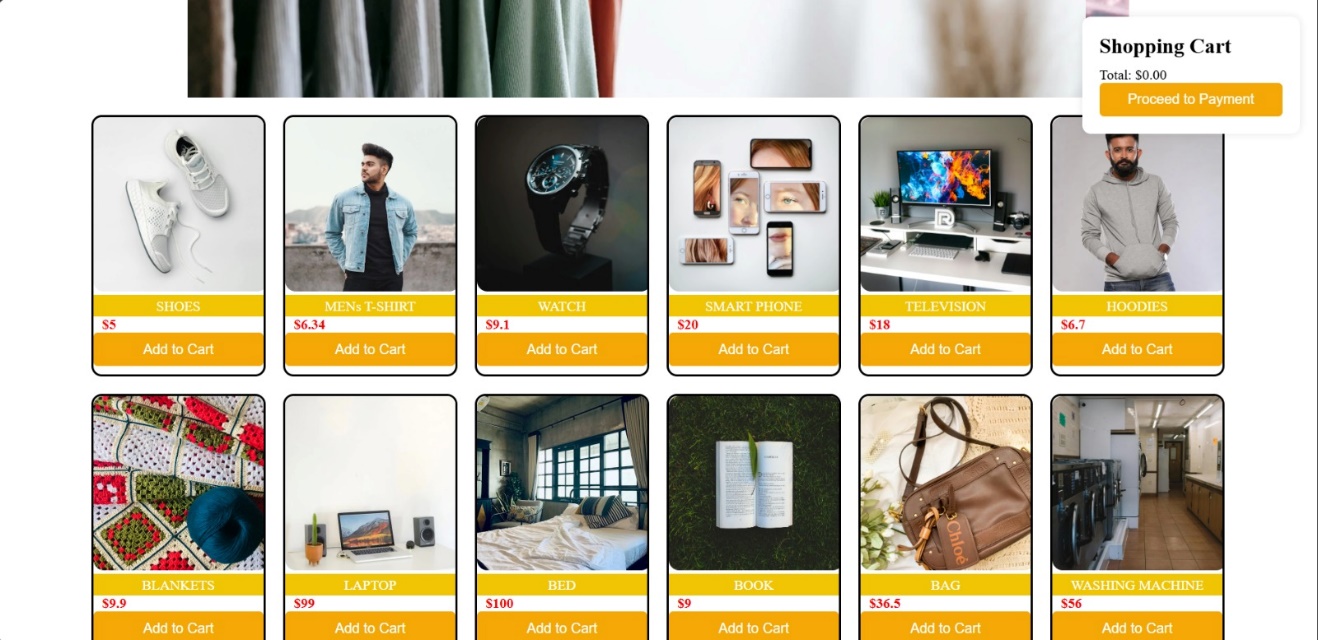
**Fig. 6 System Architecture**

A diagram of a software system

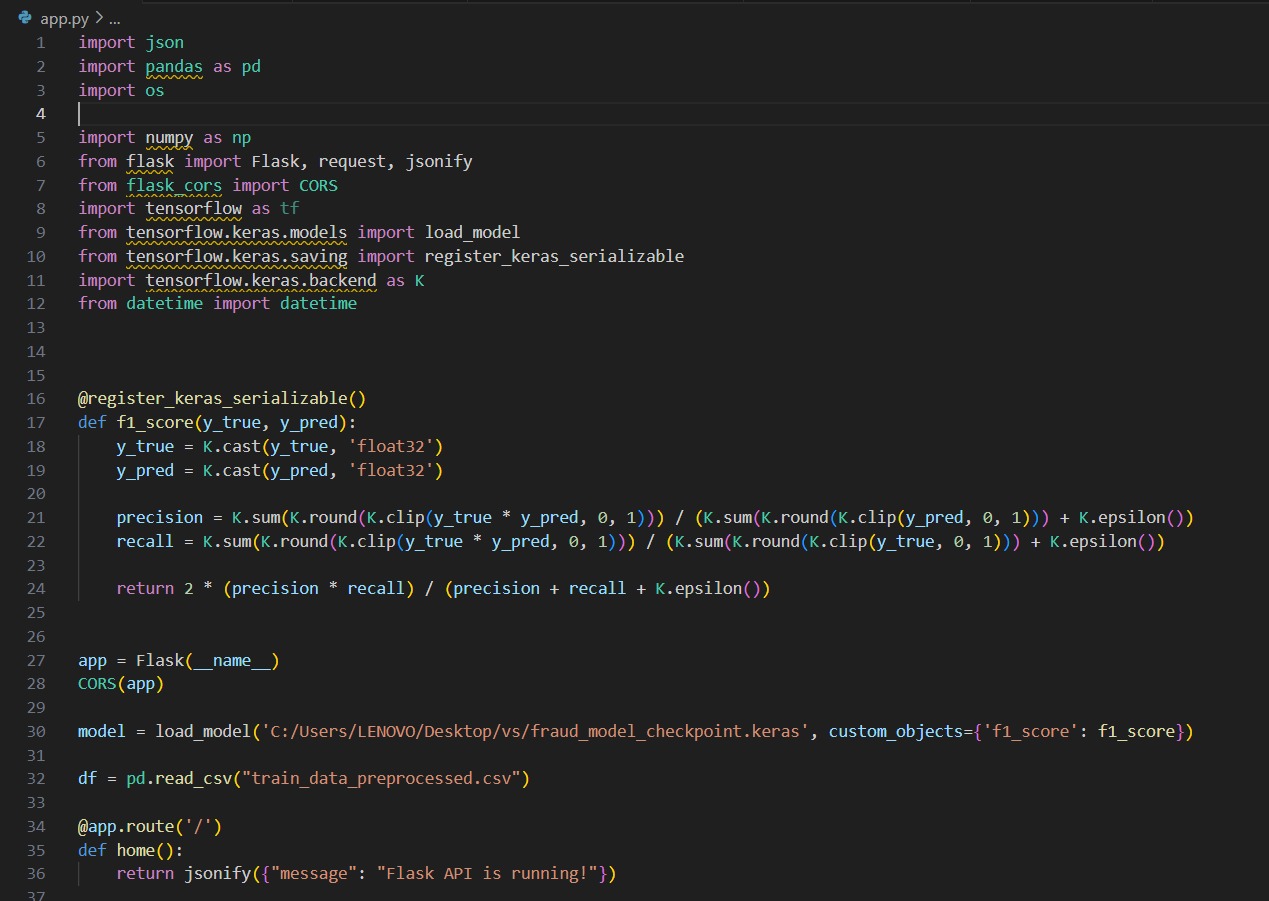
AI-generated content may be incorrect.

**Fig. 7 Website Front-End**





**Fig. 8 Website Backend**

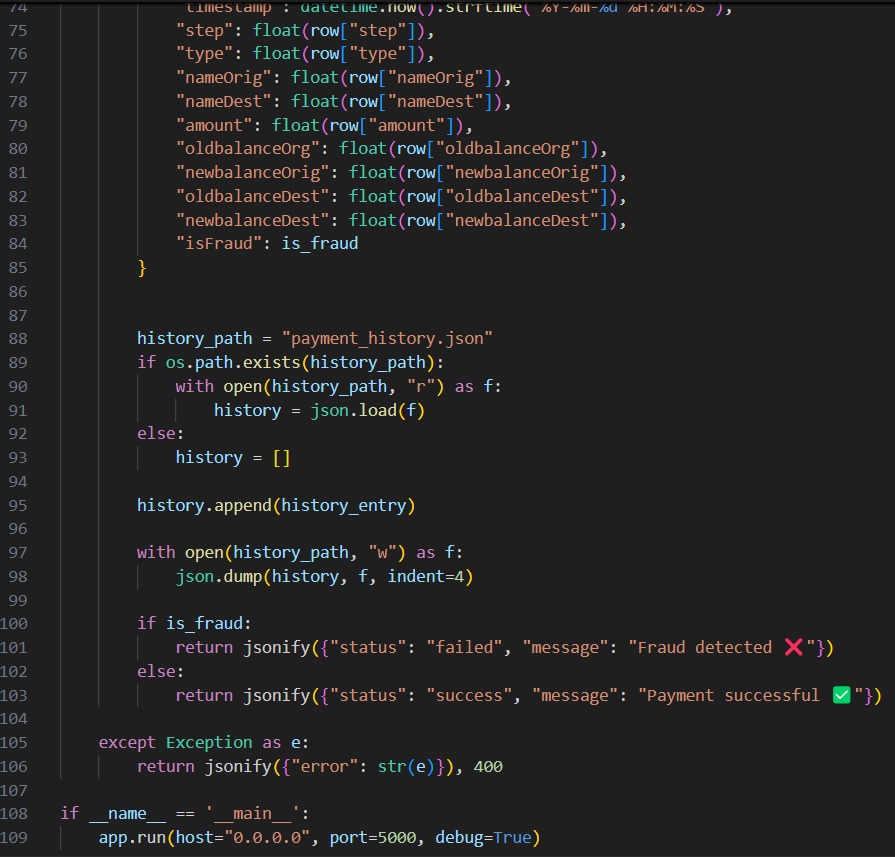


**Fig. 8.1**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Fig. 8.2**

****