İstanbul
Bilgi Üniversitesi

Credit Card Fraud detection

by

Leyana Jayousi, 120200153
Ahmed Ibrahim, 120200006
Khaled El Arabi, 120200125

Submitted to the
Faculty of Engineering and Natural Sciences
in partial fulfillment of the requirements for the

Bachelor of Science

in the
Department of Computer Engineering

January, 2024

Abstract

In credit card fraud detection, the main problem is identifying fraudulent transactions among a vast number of legitimate ones. The goal is to develop a robust model that can accurately distinguish between genuine and fraudulent transactions, thereby minimizing financial losses for both customers and financial institutions. This involves leveraging deep learning learning algorithms to analyze transaction patterns, detect anomalies, and build predictive models that can effectively flag suspicious activities in real-time.

Table of Contents

List of Figures

List of Tables

1 Introduction

In the rapidly evolving landscape of digital transactions, credit cards have emerged as a cornerstone of modern commerce, enabling swift and convenient financial interactions worldwide. However, this convenience is accompanied by a persistent threat: fraudulent activities that pose significant risks to financial institutions, businesses, and cardholders alike. The prevalence of sophisticated fraud schemes necessitates the development of advanced and adaptive solutions to safeguard against these malicious activities.

This project is an exploration into the intricate realm of credit card fraud detection, acknowledging the pressing need for robust and proactive mechanisms to identify, prevent, and mitigate fraudulent transactions. Through the fusion of data science, deep learning, and cutting-edge technologies, the primary aim is to craft a comprehensive and responsive system capable of swiftly discerning anomalous activities amidst the vast stream of legitimate transactions.

The multifaceted nature of fraudulent transactions demands a multifaceted approach. Leveraging historical transaction data, behavioral analytics, pattern recognition, and anomaly detection techniques, this project seeks to construct intelligent algorithms that continuously evolve and adapt to the ever-changing landscape of fraudulent behavior. By training models on diverse sets of features, from transaction amounts and frequencies to user behaviors and geographical patterns, the objective is to create a robust framework capable of accurately discerning suspicious activities in real-time.

2 Methodology

The choice of a feedforward neural network architecture using keras for this credit card fraud detection project was deliberate and based on several considerations. The nature of the dataset primarily consisted of tabular data comprising various numerical data. Given the structured nature of this data, a feedforward neural network using densely connected layers appeared suitable for extracting complex patterns and relationships embedded within this information.

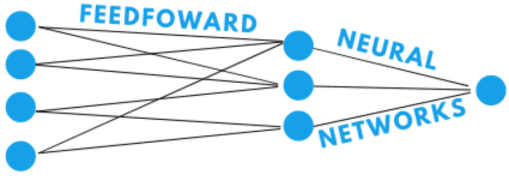
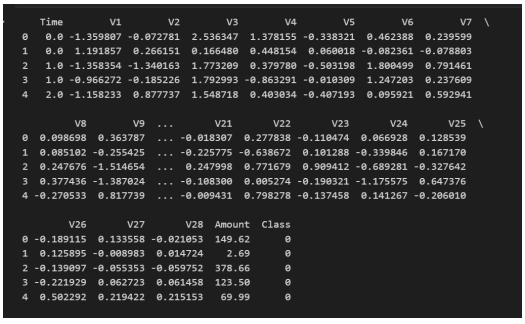


Figure 1: Simple neural network architecture

2.1 Dataset

The dataset utilized in this study stems from the Credit Card Fraud Detection.[1] This dataset presents transactions that occurred in two days, where there is 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. The dataset contains 31 columns with one dependent column which is the response variable and it takes value 1 in case of fraud and 0 otherwise.



```
   Time        V1        V2        V3        V4        V5        V6        V7  \
0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

         V8        V9  ...       V21       V22       V23       V24       V25  \
0  0.098698  0.363787  ... -0.018307  0.277838 -0.110474  0.066928  0.128539
1  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288 -0.339846  0.167170
2  0.247676 -1.514654  ...  0.247998  0.771679  0.909412 -0.689281 -0.327642
3  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321 -1.175575  0.647376
4 -0.270533  0.817739  ... -0.009431  0.798278 -0.137458  0.141267 -0.206010

        V26       V27       V28  Amount  Class
0 -0.189115  0.133558 -0.021053  149.62      0
1  0.125895 -0.008983  0.014724    2.69      0
2 -0.139097 -0.055353 -0.059752  378.66      0
3 -0.221929  0.062723  0.061458  123.50      0
4  0.502292  0.219422  0.215153   69.99      0
```

Figure 2: Dataset information

2.2 Dataset Split

The dataset was initially prepared by separating the features (X) from the target variable (y), which denotes the 'Class' indicating fraudulent or non-fraudulent transactions. To handle missing values, the dataset's missing entries were imputed with their respective column means. Subsequently, the features underwent a conversion to numeric format to ensure uniform data types. Prior to model training, the dataset was split into training and testing sets using an 80-20 ratio. To standardize the data and bring features to a common scale, a StandardScaler was applied, transforming both the training and test sets to ensure consistency in feature distributions and magnitudes, thereby optimizing the learning process for subsequent machine learning models.

2.3 Dataset balancing

As previously discussed (Fig. 2), the classes present an imbalance that should not be overlooked when training our networks, as they may develop biases towards the majority classes, thus generalizing badly to unseen data. The literature suggests various techniques to address this issue, ranging from simple approaches like undersampling or oversampling. In this work, we use a weighting mechanism to oversample minority classes. The weight of a class $\mathcal{C} \in \mathscr{C}$ is given by

$$w_{\mathcal{C}} = \frac{1}{\#\mathcal{C}} \tag{1}$$

where $\#\mathcal{C}$ represents the number of instances in class $\mathcal{C}$. The reasoning behind this is to assign higher weights to minority classes, making them more likely to be sampled during the training process. The figure below (Fig. 3) shows the imbalance and distribution of the classes after applying weighted oversampling.

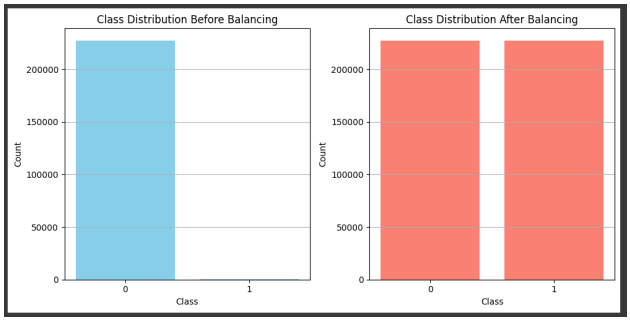


Figure 3: Class distribution before and after balancing

2.4 Model architecture

This model architecture is a Sequential neural network, a straightforward stack of layers using the Keras API. It consists of dense layers, each layer is responsible for learning and extracting different features from the input data[2]. Dropout layers are strategically placed to reduce overfitting by randomly "dropping out" connections between neurons during training. The selection of activation functions, applied to intermediate layers and the output layer, enables the model to effectively capture nonlinear relationships within the data and generate binary classification predictions. Overall, this architecture prioritizes simplicity and adaptability, utilizing dense connections and dropout regularization to create a versatile model capable of capturing patterns within the dataset for effective fraud detection.

3 Models

All models in this study underwent a standardized training regimen, each consisting of 15 training epochs with a fixed batch size of 64. This consistent approach ensured a comparable evaluation of various model architectures and configurations.

3.1 First model

The model architecture follows a Sequential neural network design comprising three dense layers. The initial layer includes 32 neurons activated by Rectified Linear Units (ReLU), followed by a dropout layer with a 0.1 rate, serving to alleviate potential overfitting during training. Subsequently, a second dense layer of 16 neurons, also activated by ReLU, is incorporated, followed by another dropout layer with a 0.1 rate. The final layer, employing the sigmoid activation function, houses a single neuron for binary classification. The 'adam' optimizer, coupled with binary cross-entropy loss, is utilized to compile the model, optimizing its parameters for improved predictive accuracy. Results are shown in Fig.(4)

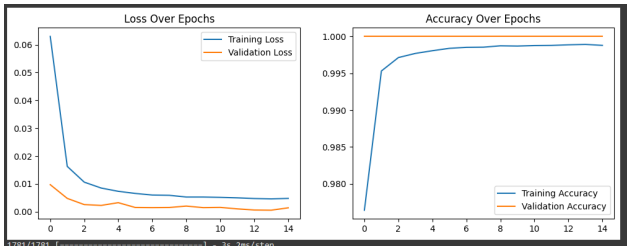


Figure 4: First experiment results

3.2 Second model

The second model adopts a Sequential neural network structure with three dense layers. It commences with an initial layer containing 64 neurons activated by Rectified Linear Units (ReLU), followed by a dropout layer with a 0.9 dropout rate, intended to minimize overfitting during training by deactivating a substantial portion of neurons randomly. A subsequent layer with 32 neurons, also activated by ReLU, is introduced, followed by another dropout layer with the same 0.9 dropout rate. The final layer, utilizing the sigmoid activation function, houses a single neuron for binary classification.

Notably, this model employs Stochastic Gradient Descent (SGD) as its optimizer, specifically configured with a learning rate of 0.01, and uses binary cross-entropy loss as its optimization metric. Results are shown in Fig.(5)

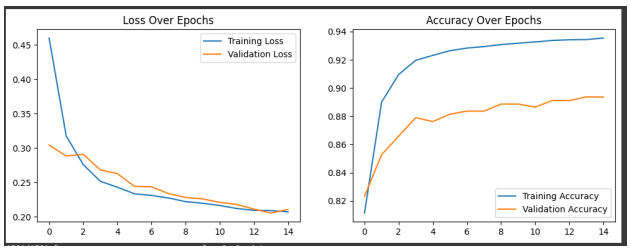


Figure 5: Second experiment results

3.3 Third model

The third model architecture comprises a multi-layered neural network structured with diverse layers employing hyperbolic tangent (tanh) activation functions. This architecture starts with an input layer of 128 neurons, followed by subsequent layers employing a dropout rate of 0.5 to reduce overfitting. The intermediate layers consist of a dense layer with 64 neurons, a more extensive layer with 500 neurons, another layer with 64 neurons, and a final layer of 128 neurons before the output layer with a single neuron employing the softmax activation function for binary classification. The model is compiled using the Adam optimizer and binary cross-entropy loss, with an evaluation yielding test loss and accuracy metrics.Results are shown in Fig.(6)

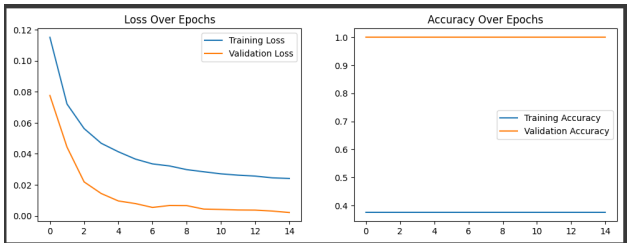


Figure 6: Third experiment results

4 Results

The evaluation of different neural network architectures for credit card fraud detection highlighted varying performance outcomes. While each model showed promise in classifying fraudulent transactions, the presence of a class imbalance affected their ability to generalize. Deeper architectures exhibited higher accuracy but struggled with biases toward the majority class, potentially missing minority class instances. The findings emphasize the need to address class imbalances for more robust and equitable fraud detection models.

The table below (Table 1) showcases the accuracy outcomes obtained from training distinct Keras models using various optimization algorithms and hyperparameters. These results offer valuable insights into the nuanced impact of optimization techniques on the performance and convergence of the models.

Experiment	Parameters					Marices						
	Batch size	Epochs	Optimization alg.	no. of neuron in each layer	Dropout%	Precision		Recall		F1-score		Accuracy
						0	1	0	1	0	1	
1	64	15	Adam	32,16,1	0.1	1	0.57	1	0.84	1	0.68	0.99
2	64	15	SGD	64,32,1	0.9	1	0.57	1	0.84	1	0.68	0.935
3	64	15	tanh	128,64,500, 64,128,1	0.5	0	0	0	1	0	0	0.375

Table 1: Model Comparison and Accuracy

The higher accuracy achieved by the first model could be attributed to its relatively simpler architecture, featuring fewer layers and neurons. This simplicity might have contributed to less overfitting during training compared to the other models. Additionally, the careful utilization of dropout layers in the initial and intermediate layers helped in regularizing the model, preventing it from learning noise in the data and enhancing its generalization capabilities. Furthermore, the use of the 'adam' optimizer might have facilitated quicker convergence and better parameter optimization, leading to higher accuracy in this particular scenario.

5 Conclusion

The investigation into neural network architectures for credit card fraud detection highlighted the significant influence of class imbalance on model performance. While the models demonstrated varying degrees of accuracy, the presence of imbalanced data noticeably affected their predictive capabilities. Deeper and more complex architectures exhibited potential but faced challenges in achieving balanced predictions, showcasing a bias towards the majority class. The findings underscore the critical need to address class imbalance in fraud detection tasks, as it impacts the model's ability to generalize and effectively identify fraudulent transactions. Future studies should prioritize techniques to mitigate class imbalance, explore alternative architectures, and consider advanced methodologies to enhance the robustness of fraud detection models in real-world scenarios.

References

[1] U. L. de Bruxelles, "Credit card fraud detection." https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud, 2021.

[2] L. Denoyer and P. Gallinari, "Deep sequential neural network." https://arxiv.org/pdf/1410.0510.pdf, 2014.