POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Credit Card Fraud detection - NAML project

Author(s): **Federica Topazio 10985760**

Academic Year: 2023-2024

Contents

1 | Scope

1.1. Introduction

The referenced paper introduces the use of deep learning methods, which are still very limited in use in the area of credit card fraud detection. The referenced networks then used in the paper are two different types of CNN and LSTM, since these are the ones more suitable for large datasets. Another big impact on the performance of the classification mechanism is done by the data pre-processing in credit card fraud. However, how this step is performed is not explicitly unveiled in the paper.

Another main focus of this study is to determine the performance of classifiers for credit card fraud detection on datasets having a varied number of samples and features. For this purpose, three different datasets, i.e. European Card Data (ECD), Small Card Data (SCD) and Tall Card Data (TCD) are used. Like all credit card fraud datasets where fraud instances are very less compared to normal transactions, these datasets are highly imbalanced. So, imbalance needs to be addresses in the data pre-processing.

All three datasets in this study are labelled datasets with class value '0' representing no fraud and '1' indicating fraud. (Class = target variable)

1. *European Card Data*: This dataset contains 284,807 samples and 31 features. Out of the given samples, only 492 are fraud cases and account for 0.172% of the dataset. Due to the privacy of customer information and the sensitivity of transactional details, all except 'Time' and 'Amount' features in the dataset are PCA transformed. The 'Time' feature represents the time in seconds passed starting from the first sample in the dataset and the 'Amount' feature shows the total amount of the transaction. This dataset is referred to as 'ECD'.

2. *Small Card Data*: This dataset is a small dataset containing 3075 samples and 12 features. Half of the features are categorical while another half is numerical. Out of 3075 samples, 448 are fraud cases contributing to 14.6% of all cases. The features used in this dataset are Merchant ID, Transaction date, Average transaction amount

per day, the Transaction amount is declined, Number of declines per day, is a foreign transaction, is a high-risk country, Daily chargeback average amount, six-month chargeback average amount, six-month chargeback frequency and is fraudulent. Due to a smaller number of rows and columns, this data set is named as Small Card Data (SCD).

3. *Tall Card Data*: This dataset has a high number of samples and a low number of features this dataset is named Tall Card Data (TCD) in this study. Only 5.96% of the dataset contains fraud cases. The features comprising the dataset are customer ID, gender, state, number of cards a customer has, balance on the card, number of transactions to date, number of international transactions to date, customer's credit line and fraud risk indicating fraud or non- fraud. Due to limited computing power and higher training times associated with the classifiers the paper considers only a small proportion of this data for the study.

Unfortunately, the site where it was supposed to be retrieved the last dataset does not respond, and it will not be further mentioned in this report.

1.2. Aim

As this paper reports the results accomplished on the different datasets, considering different neural networks and techniques to balance the dataset, the aim of this project is to try to recreate such results, drawing conclusions from the obtained numbers.

2 | Main Aspects

Credit card fraud is an increasing phenomenon posing a significant challenge to the financial industry. Due to these fraudulent activities, consumers are becoming wary of making purchases, while merchants and financial institutions are suffering substantial financial losses. Key issues in tackling credit card fraud include the availability of public data, the severe imbalance in data classes, the evolving nature of fraud, and the high incidence of false positives. Although machine learning techniques have been employed to detect credit card fraud, none have achieved high efficiency so far. Recently, deep learning has been utilized to address complex problems across various domains. Therefore, the goal will be to try reproducing the results achieved in the paper, analysing them and drawing conclusions from it.

2.1. Evaluation metrics

To evaluate the performance of the models that will be analysed, it is used a combination of accuracy and recall.

Confusion Matrix

The confusion matrix is a table used to evaluate the performance of a classification models. It is defined as follows:

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Where:

- TP: True Positives
- TN: True Negatives

- FP: False Positives
- FN: False Negatives

Accuracy

Accuracy is the ratio of correctly predicted instances to the total instances.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Recall (Sensitivity)

Recall, also known as Sensitivity or True Positive Rate, is the ratio of correctly predicted positive observations to the actual positives.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positives.

$$\text{Precision} = \frac{TP}{TP + FP}$$

F1 Score

The F1 Score is the harmonic mean of Precision and Recall, providing a balance between the two.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Combining Accuracy and Recall

A comprehensive evaluation of model performance often requires a balance between accuracy and recall, since accuracy alone is not used for evaluation because of the data imbalance in credit card frauds. The recall ensures the robustness of the system.

2.2. Imbalanced Classification Techniques

Class imbalance is due to the unequally distribution of samples in a labeled dataset, resulting in majority and minority class. Credit card transactional data is particularly imbalanced, as fraudulent transactions are very few compared to the millions of legitimate transactions processed daily. An imbalanced dataset, where the minority class is the class of interest, can lead to poor performance by biasing towards the majority class and misclassifying the minority class as noise. Therefore, in this paper and I have chosen to use on all datasets analysed, the following imbalanced classification techniques.

Random Under-Sampling

Random under-sampling aims to balance the class distribution by randomly removing instances from the majority class. This can be defined as follows:

Given a dataset $D = \{(x_i, y_i)|i = 1, 2, \ldots, n\}$, where $y_i \in \{0, 1\}$ and the majority class is labeled as 0, random under-sampling reduces the dataset size by removing instances from the majority class.

Let n_0 and n_1 represent the number of instances in the majority and minority classes, respectively, with $n_0 > n_1$. The under-sampled dataset D' will have $n'_0 = n_1$ instances from the majority class and n_1 instances from the minority class:

$$D' = \{(x_i, y_i)|y_i = 1\} \cup \{(x_j, y_j)|y_j = 0, j \in \text{RandomSample}(n_0, n_1)\}$$

NearMiss

NearMiss is a type of under-sampling technique that selects majority class instances based on their distance to minority class instances. Specifically, NearMiss-1 selects majority class instances with the smallest average distance to the three closest minority class instances.

Mathematically, let $d(x_i, x_j)$ denote the distance between instances x_i and x_j. For each majority class instance x_i, compute the average distance to the k nearest minority class instances:

$$\text{AvgDist}(x_i) = \frac{1}{k} \sum_{j=1}^{k} d(x_i, x_j^{(min)})$$

NearMiss-1 selects the majority class instances with the smallest $\text{AvgDist}(x_i)$ values.

SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE generates synthetic examples for the minority class to balance the dataset. The algorithm works as follows:

1. For each minority class instance x_i, find its k-nearest neighbors.
2. Randomly select one of the k-nearest neighbors, $x_i^{(nn)}$.
3. Generate a synthetic instance $x_{synthetic}$ as a convex combination of x_i and $x_i^{(nn)}$:

$$x_{synthetic} = x_i + \lambda(x_i^{(nn)} - x_i)$$

   where $\lambda \in [0, 1]$ is a random number.

Given a dataset $D = \{(x_i, y_i)|i = 1, 2, \ldots, n\}$, where $y_i \in \{0, 1\}$ and the minority class is labeled as 1, SMOTE increases the number of minority class instances by adding synthetic examples.

3 | Methodology

As mentioned in the previous steps, the analysed datasets will be ECD and SCD. The first dataset is available at the following link: ECD dataset link, whereas the second one is available at: SCD dataset link.

In the notebook, the aforementioned datasets will not be downloaded locally, but they will be retrieved dynamically from kaggle inserting the json token of the kaggle account of whoever is executing the code.

3.1. Data Pre-processing

The initial step is data pre-processing. This involves a thorough exploration of all the two datasets by manually inspecting them and to refine the input to the classifiers to ensure optimal output. Factors such as missing values, categorical features, variable scales, and high dimensionality can all impact classifier performance. The two main pre-processing methods utilized are data exploration, data scaling, and test-train splitting.

- For the ECD dataset, all features were numeric, there were no missing values, and no features were dropped during data cleaning.
- In the SCD dataset, all categorical features were converted to numeric, and the 'Transaction Date' feature was dropped due to all values being missing.

Both the ECD and SCD datasets were used in their entirety.

Correlation analysis

Identifying correlations can help eliminate features with similar behavior, thereby reducing the dimensions of the data. Lower-dimensional data can improve training times and classification performance.

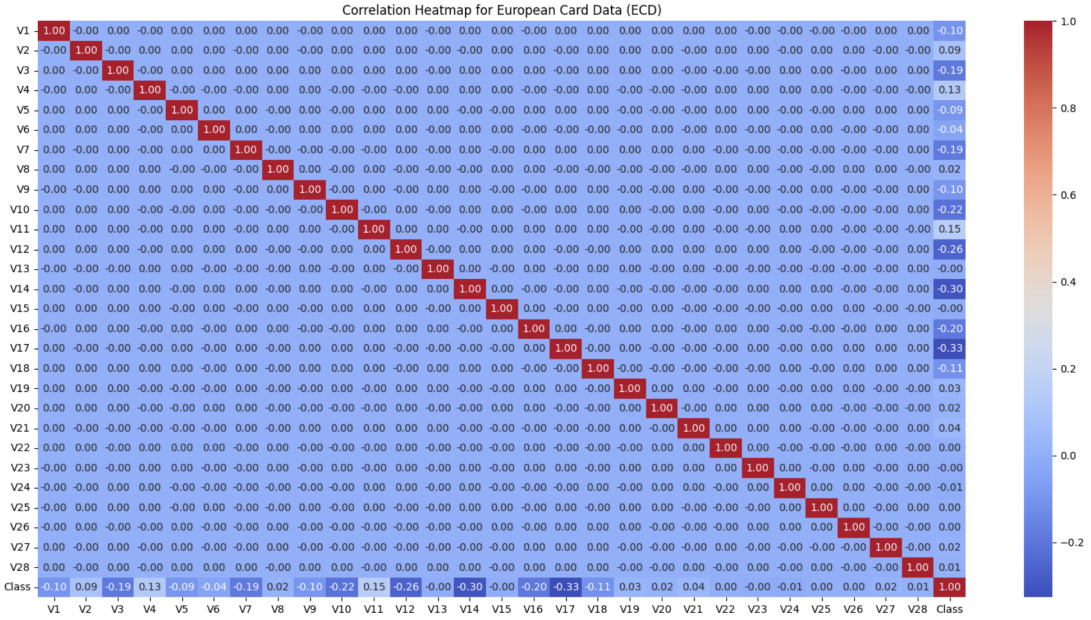

Figure 3.1: Correlation matrix - ECD dataset

For the ECD dataset, most features exhibit very low or near-zero correlations with each other. This indicates that the features in the ECD dataset are largely independent of one another, providing unique and diverse information. It can be observed that there is no negative correlation between the features and no uniform correlation throughout the SCD dataset. A higher average number of transactions per day is associated with higher transaction amounts. The daily chargeback amount, six-month chargeback amount, and six-month chargeback frequency are also highly correlated with each other. Lastly, the 'high risk country' feature is most significant in determining the fraud class. Since this dataset is already very small, no dimensionality reduction is performed.

3.2. Data scaling and standardization

Some features within the datasets have different scales. For instance, in the ECD dataset, the 'Amount' feature has a mean of 94826.6, while the 'V1' feature has a mean of 0.000639. Scaling and standardization techniques adjust these features to a similar scale, making the data more interpretable for the Deep learning algorithms. In this study, the Stan-

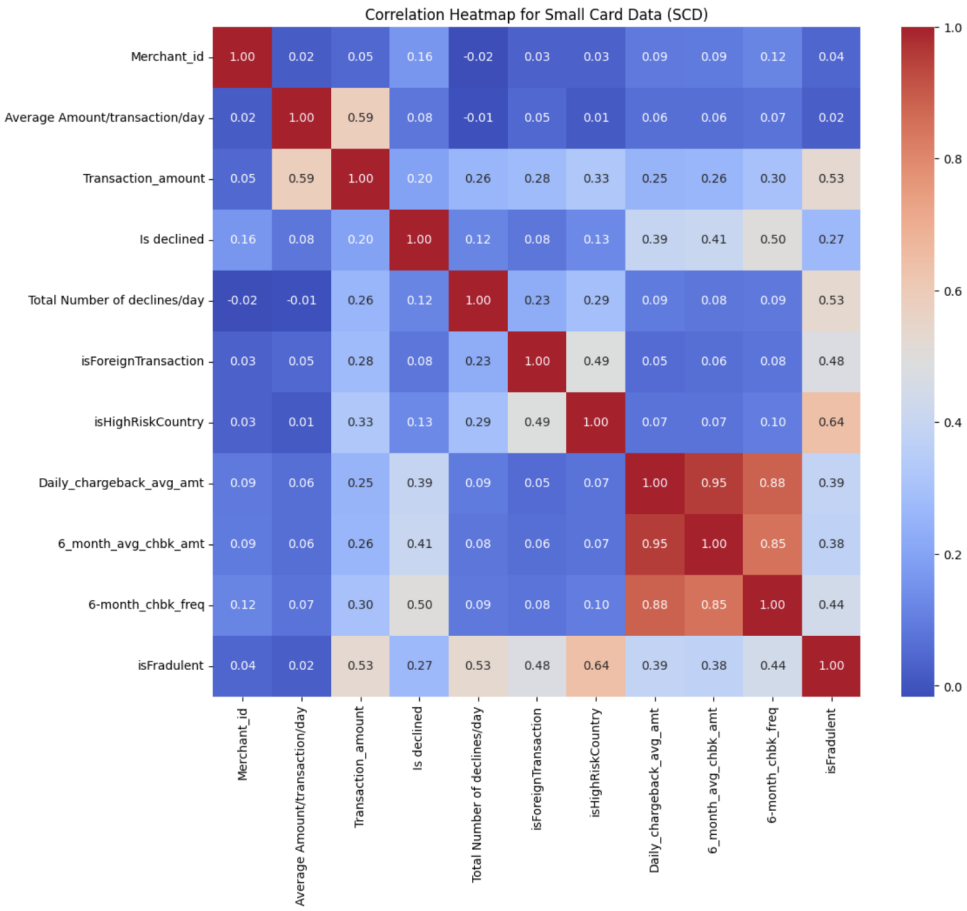

Figure 3.2: Correlation matrix - SCD dataset

dardScaler class from the sklearn pre-processing library in Python is utilized. The StandardScaler transforms each feature such that the mean is 0 and the standard deviation is 1. After applying this scaler, the means of the aforementioned features are both adjusted to 0.

3.3. Test, Train, and Validation Split

The three datasets are divided into test, training, and validation sets. The test data comprises a small random sample, constituting 3.5% of each dataset. The remaining data is split into training and validation sets, with 80% allocated for training and 20% for validation. The training data is used to train the models, while the validation data helps to fine-tune the classifiers and prevent overfitting, thereby enhancing model performance.

3.4. Models

How the deep learning methods such as LSTM and two kinds of CNN perform for credit card fraud classification is the major focus of this study.

One-Dimensional CNN (1DCNN)

Convolutional Neural Networks (CNN) are deep learning methods extensively used with spatial data, particularly in image processing. Similar to Artificial Neural Networks (ANN), CNNs have hidden layers but also include convolutional layers with varying numbers of channels in each layer. Convolution involves using moving filters to capture key information from the data. The primary goal of using CNNs in image processing is to minimize data dimensions while preserving essential features for accurate predictions.

CNN layers are not fully connected, and maintain constant weight parameters for each filter. These characteristics significantly reduce the number of parameters in a CNN model.

Typically, CNN models are designed for image and video processing. A common application of 1D CNNs is in Natural Language Processing (NLP), a sequence classification problem. In 1D CNNs, the kernel filter moves vertically through a sequence of data samples, unlike in 2D CNNs where it moves horizontally and vertically.

1D CNN Structure

Layer 1: Input

- Input Shape: (Row sample, 1, Number of Features)

Layer 2: CONV1D Layer

- Number of channels: 64
- Kernel Size: 1
- Activation Function: ReLU

Layer 3: CONV1D Layer

- Number of channels: 64

- Kernel Size: 1
- Activation Function: ReLU

Layer 4: Dropout

- Threshold: 0.5

Layer 5: MaxPooling1D

- Pool size: 1

Layer 6: Flatten

- Number of Nodes: 64

Layer 7: Dense

- Number of Nodes: 100
- Activation Function: ReLU

Layer 8: Output

- Number of Nodes: 1
- Activation Function: Sigmoid

2D CNN

Layer 1: Input

- Input Shape: (Row sample, 5, 6, 1)

Layer 2: CONV2D

- Number of channels: 64
- Kernel Size: 3x3
- Activation Function: ReLU

Layer 3: CONV2D

- Number of channels: 32
- Kernel Size: 3x3
- Activation Function: ReLU

Layer 4: Flatten

- Number of Nodes: 64

Layer 5: Output

- Number of Nodes: 1
- Activation Function: Sigmoid

LSTM

Long Short Term Memory Network

Long Short Term Memory (LSTM) is a type of Recurrent Neural Network (RNN). Standard neural networks cannot retain previous information, requiring them to relearn from scratch each time. However, RNNs often suffer from short-term memory due to the vanishing gradient problem. LSTMs address this short-term memory issue by maintaining a cell state, which serves as the network's memory across each step. Gates at each step control the memory flow by retaining necessary information and discarding irrelevant data. The gates include the forget gate, input gate, and output gate. The forget gate determines what information to retain from the previous step, the input gate decides what new information to add, and the output gate sets the hidden state for the next step. The hidden state retains information from previous inputs but still suffers from short-term memory, whereas the cell state overcomes this by remembering key information from the earliest examples in the sequence.

The LSTM network is structured as follows:

- The input layer has a shape of (1, number of features).
- The second layer is a dense layer with 50 LSTM blocks and uses the ReLU activation function.

- The output layer consists of a single node with a sigmoid activation function.

3.5. Workflow

The notebook is structured, starting from the retrieval of the datasets, followed by the pre-processing, and after which each model is trained on both datasets.

Furthermore, all models are retrained on each new dataset obtained performing on each every imbalanced classification technique exposed in the previous chapter.

4 | Results and conclusions

Anomaly detection datasets, as it has been overly discussed in this report are very imbalanced. However, one of the two datasets used (SCD), despite being imbalanced itself, it has a lower dimensional order than the other (ECD), therefore its related results could be considered trivally more significant with respect to the others, even without the use of sampling. In the training procedure, a good technique that it has been decided to be employed is using a higher number of epochs and employing callback technique, which are strategies to enhance the training process and improve the model's performance. Training a neural network for more epochs allows the model to learn better representations from the training data. Each epoch is an opportunity for the model to adjust its weights and reduce the loss.

Early Stopping is a technique where the training is stopped early if the model's performance on a validation set starts to degrade, in this case validation loss, in order to prevent overfitting by stopping the training once the model starts to overfit the training data and perform worse on validation data.

Another technique that could be employed is the ReduceLROnPlateau, which is a learning rate scheduler in Keras that reduces the learning rate when a metric has stopped improving. It could be particularly useful for fine-tuning the learning rate during training, which can help in dealing with overfitting, especially in anomaly detection where there is an high imbalance in the datasets.

The results obtained in the notebook closely match those reported in the paper. However, it's noteworthy that using NearMiss results in very low precision. This discrepancy is likely due to the data pre-processing steps, which are not fully disclosed in the paper. Therefore, the pre-processing was conducted based on the closest possible interpretation of the available information.

To enhance the results further, one potential approach is to simplify the network and incorporate regularization techniques, such as ridge regression with an L2 norm, to reduce overfitting.

List of Figures