Reproducibility in ML

Credit Fraud Detection using LSTM and Autoencoder

Mini project 4 – COMP 551 (Winter 2022)

1 INTRODUCTION

Nowadays, the usage of credit cards in everyday transactions has replaced the cash payment methods; however, as technology has progressed, the risk of credit card fraud has increased significantly. According to the Nilson Report site, in 2022, money lost in credit card fraud has reached 28.58 billion dollars. It is expected that the probability and amount of the reported frauds will increase in the following years. Therefore, machine learning developers have tried to come up with fraud and anomaly detection algorithms to help decrease the possible fraudulent transactions.

2 SCOPE OF REPRODUCIBILITY

In this assignment, we focused on implementing two articles:

(1) Enhanced credit card fraud detection based on attention mechanism and LSTM deep model
(2) Credit card fraud detection using autoencoder model in unbalanced datasets

The first article focuses on using the Long Short-Term Memory (LSTM) model. It suggests an attention layer as a possible modification that can help with model recall and make the classifier more capable of finding global fraud transactions. The second study claims that autoencoders can detect fraudulent transactions without the need for data augmentation or other complex data processing.

In the first case, we tried to implement their proposed model and study the effect of feature selection and feature extraction on the overall performance of both LSTM and LSTM + attention models; we also examined the importance of data augmentation. In the second case, we reproduced the results using autoencoder architecture and tried other architectures to see how their performance differ from the reported model.

3 METHODOLOGY

3.1 DATASET AND PREPROCESSING

The dataset used in this experiment is the credit card fraud detection dataset provided by Kaggle. This dataset contains transactions made for two days via credit cards in September 2013 by European cardholders. It includes 492 fraudulent and 284,315 normal transactions (fraudulent transactions are 17.27% of all reported transactions that happened during that period). This dataset contains only numerical features resulting from a PCA transformation. Note that dataset publishers have mentioned in their dataset description that they could not provide the original features themselves because of confidentiality issues. Hence, there are 28 features (V1, V2, … V28) resulting from applying PCA plus two features called 'Time' and 'Amount' that have not been transformed with PCA. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset, and the feature 'Amount' is the transaction amount. Finally, 'Class' is the response variable, and it takes the value of 1 in case of fraud and value of zero otherwise. **Figure 1** shows data distribution per class label.

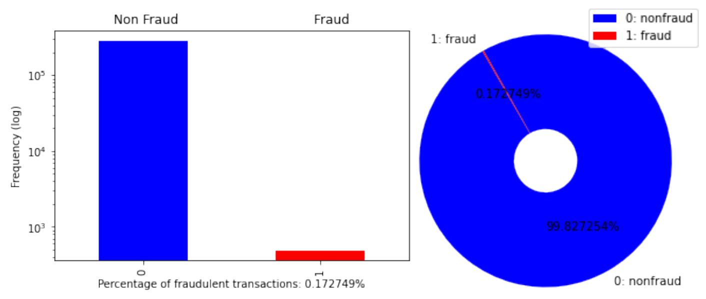


Figure 1 Credit card fraud detection dataset is highly imbalanced.

Here we tried to check if we needed data cleaning. Hence, we checked for missing values or duplicated entries and found no issue. The only significant observation is that the dataset is highly imbalanced, implying that any classification algorithm will be biased towards the large categories. Hence, it hints not to use accuracy as the measure of classification performance when discussing and comparing the models; instead, studying measures such as recall or Area Under Curve (AUC) can be more informative. It also suggests the idea of data augmentation using algorithms such as the Symmetric Minority Over-Sampling Technique (SMOTE). The SMOTE algorithm can augment the minority class data while not affecting the distribution of features. This method uses the KNN algorithm to create new synthetic samples to balance the class distribution of the dataset (Safdari et al., 2022; Chawla et al., 2002). The first article has used this data augmentation strategy as a part of its preprocessing steps (no augmentation in the second dataset). The following figures show the V1 vs. V2 values distribution before and after applying SMOTE; **Figure 2** shows no change in data distribution after using this methodology.

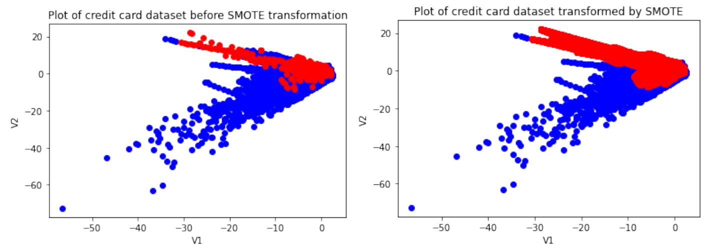


Figure 2 Right panel: V2 vs. V1 features before applying SMOTE. Left panel: V2 vs. V1 features after using SMOTE

3.2 MODEL DESCRIPTION

The main model in the first article is called LSTM, a type of Recurrent Neural Network (RNN) capable of processing time sequences of data. In contrast to standard feedforward neural networks, LSTM has feedback connections between hidden units associated with discrete time steps. This helps with understanding the dependencies between the time steps in sequenced data. These networks are widely used in DL and NLP tasks like speech recognition, stock market prediction, and anomaly detection. In this article, the authors have suggested using an attention layer plus the LSTM model. The attention mechanism allows a sequence-based neural network to automatically focus on the critical data items for the classification task. The attention mechanism works by taking a weighted average of a series of vectors to create a context vector with the most important information, which is then utilized as input in the following layer.

In the second article, an autoencoder model is implemented. Autoencoders are a type of unsupervised artificial neural network used for representative learning. Encoder and decoder are the two main blocks of Autoencoders. The encoder part compresses the input data to a bottleneck known as the latent space representation. These encoders are followed by some decoder layers that try to reconstruct the input from the compressed bottleneck. In this way, the autoencoder learns a representation (encoding) of the input data used for feature extraction or dimensionality reduction tasks. Autoencoders can be used to detect credit card fraud as it is a type of anomaly detection. Autoencoders aim to learn the features of normal transactions for which the input will be similar to the output reconstructed. On the other hand, the input and output will vary for fraud transactions as it is unexpected data.

3.3 Experimental setup and code

The authors of the first paper have released the source code and results of the proposed model at https://github.com/bibtissam/LSTM-Attention-FraudDetection.

For the second dataset, the authors themselves have not published the code they have used, but we tried to implement it and see how autoencoders could work on the credit fraud detection problems.

4 Results

In this project, the first aim was to study the model presented by Benchaji et al. called "Enhanced credit card fraud detection based on attention mechanism and LSTM deep model". Their article mentions that they have done feature extraction using UMAP algorithm before introducing the data to their proposed model; however, we could not find the feature extraction part in their released code. Hence, we tried to implement this step by ourselves (part 4.1.1); we also ran the models without this preprocessing step (as the code suggested) and studied the models' performance in that scenario (part 4.1.2).

4.1.1 LSTM/LSTM + attention and UMAP (with feature extraction)

As mentioned in the article, feature selection and dimension reduction are made before model development. Only essential features whose empirical distributions have the most significant discrimination when comparing fraudulent transactions and normal transaction classes were considered. These features would be V1, V2, V3, V4, V10, V11, V12, V14, and V17 (for more info about the empirical distribution of these features, please refer to the supplementary part of the report). Then, we used dimension reduction algorithms, including PCA, tSNE, and UMAP, to do feature extraction.

But the question is, which feature extraction method works better? To answer this question, as the article suggested, we tried to come up with a way to visualize the result of each feature extraction methodology. However, as the number of instances increases, the time required to run the feature extraction procedure increases significantly; hence we decided to decrease the number of instances when applying these methods. In this regard, we used SMOTE and sampling together (while we tried to keep the ratio of fraud vs. normal cases as ¼ and decreased the overall instances to be equal to 1178); then, we applied PCA, tSNE, and UMAP to these instances, while setting the output number of components to be equal to 3 and visualized the results per class (**Figure 3**).

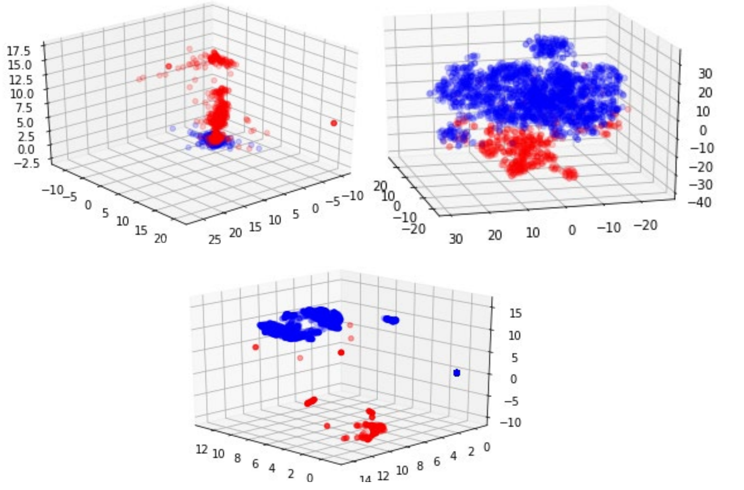


Figure 3 The performance of different dimensional reduction algorithms on credit card fraud dataset. Top left: PCA. Top right: tSNE. and bottom: UMAP

Here, one can conclude that PCA does not have a good performance on discriminating the instances of two class labels, while UMAP and tSNE work better in that term. When it comes to comparing tSNE and UMAP, UMAP seems to be a better option than tSNE both in terms of providing discrimination and runtime. Hence, for the rest of the method development, the idea would be to apply UMAP to the data and then apply the LSTM and the LTSM + attention models to the data to do the classification. However, to avoid high computational load and time efficiency, we used SMOTE, sampled from the instances (½ of instances are chosen), applied UMAP, and fed the resulting components into the model. Note that in this section, the dimension of the space to embed into was six. Here, we have presented the confusion matrixes of the LSTM and the LSTM + attention models (considering the threshold of 0.3).

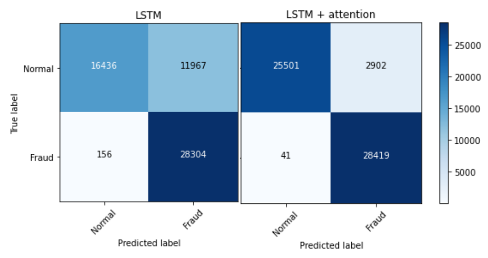


Figure 4 LSTM and LSTM + attention, both feature selection and UMAP feature extraction are included

Model	Accuracy	Recall	Precision
LSTM	78.60	99.45	70.28
LSTM + attention	**94.82**	**99.85**	**90.73**

Table 1. Comparing the performance of LSTM without and with the attention layer; UMAP feature reduction included

The Area Under Curve (AUC) clearly shows that while both models are working well in fraud detection, the AUC of the LSTM + attention model seems to be better; therefore, the proposed model of the articles seems to improve the classification problem.

Model	AUC
LSTM	0.9519
LSTM + attention	**0.9923**

Table 2. Comparing AUC of LSTM without and with the attention layer; UMAP feature reduction included

4.1.1.1 Changing a hyperparameter in LSTM layers

As illustrated above, the attention layer at the top has helped with model performance (in terms of AUC). Here, we tried to change a hyperparameter of this model to see how it can affect the model's performance. In the previous section, the number of neurons per LSTM layer was equal to 50, and now, we try to change this number to 20 and 70. It seems like having 20 neurons per layer can slightly improve the model.

Model	Neurons	AUC
LSTM + attention	50	0.9923
LSTM + attention	20	0.9975
LSTM + attention	70	0.98176

Table 3. Comparing AUC of LSTM + attention model when a hyperparameter is changed.

4.1.2 LSTM/LSTM + attention / no feature extraction

In this section, we plan not to include UMAP feature extraction step and run the code that the authors have released.

4.1.2.1 ABLATION STUDY

We used ablation study to validate the effect of data preprocessing, feature selection, and attention mechanism on the performance of fraud detection. We compared the accuracy, recall, and AUC under different conditions. Please note recall is especially important in this fraud detection task as we do not want any fraudulent transactions to be classified as normal.

4.1.2.2 ATTENTION MECHANISMS

Attention Mechanisms can integrate global information and stress the importance of features that are important for fraud detection. After adding the attention layer, the authors reported that their model got higher accuracy and recall. To validate the effect of the attention mechanism, we compared the performance with/without the attention layer.

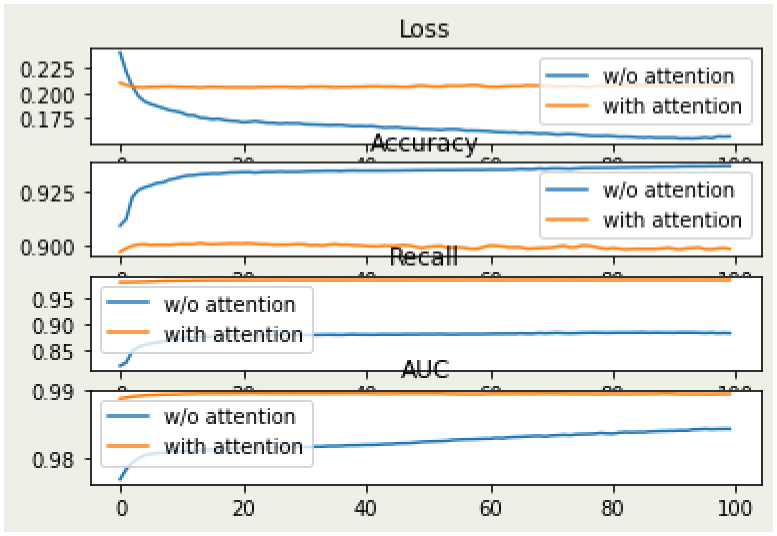


Figure 5 Comparing the performance of LSTM without and with the attention layer during training

Figure 5 shows the performance with/without attention layer on the test set during training. We can see that the model with the attention layer has higher recall and AUC but with lower accuracy and higher loss. **Figure 6** shows the confusion matrix of LSTM and LSTM-attention after 100 epoch training, which indicates the same thing. We also summarized the accuracy, recall, and AUC in **Table 4**.

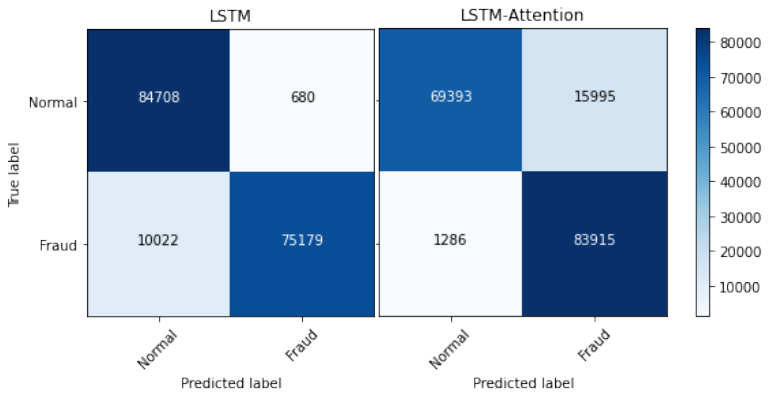


Figure 6 The confusion matrices of LSTM without and with the attention layer

	Accuracy	Recall	AUC
LSTM	0.937	0.882	0.984
LSTM + attention	**0.899**	**0.985**	**0.989**

Table 4. Comparing the performance of LSTM without and with the attention layer

Furthermore, when we compare the results obtained from this section with the results of part 4.1.1 where we had added the feature extraction step (**Table 2**), we can see that applying UMAP to the data seems to improve AUC. This observation suggests feature extraction using UMAP as a good preprocessing strategy.

4.1.2.3 DEAL WITH DATA UNBALANCE WITH SMOTE

To deal with the severe class imbalance of the credit card fraud detection dataset (with 284315 normal transactions and only 492 fraud transactions), the authors used SMOTE to generate synthetic samples of the minority class. We also compared the results of the model with and without SMOTE. The confusion matrices are shown in **Figure 7,** and we can get the accuracy, recall, and AUC from these confusion matrices shown in **Table 5**. As we can see, after using SMOTE, the recall increases from 0.779 to 0.985, and the AUC increases from 0.912 to 0.989, which illustrates that data augmentation is significant when data is imbalanced.

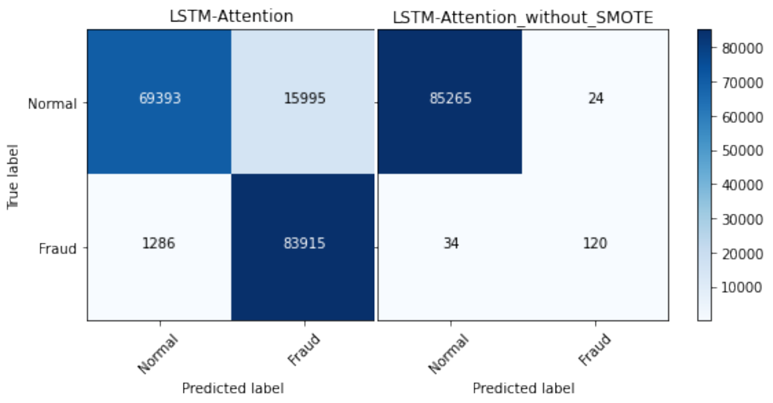


Figure 7 The confusion matrices of LSTM + Attention with and without SMOTE

	Accuracy	Recall	AUC
LSTM + Attention SMOTE	0.899	**0.985**	**0.989**
LSTM + Attention _without_SMOTE	0.999	0.779	0.912

Table 5. Comparing the performance of LSTM + Attention with and without SMOTE

4.1.2.4 FEATURE SELECTION

The original dataset had 30 features, and the authors used feature selection to remove redundant and unrelated features and finally kept 9 features as input. To validate the effect of feature selection, we compared the results of the network using the original 30 features and nine selected features as input. **Figure 8** shows the test set's performance during training with and without feature selection. As you can see, the loss, accuracy, and AUC of models without feature selection become better than that of the model with feature selection after epoch 6. And the recall of the model without feature selection becomes very similar to that of the model with feature selection at epoch 100. We also showed the confusion matrices in **Figure 9** and summarized the metrics in **Table 6**. According to our experiment, although it will bring some unrelated and redundant information if we input all the features, the accuracy and AUC are better than the model with feature selection. Their results of recall are comparable. And please note that including more features does not increase the training time. Including all features without feature selection can be better.

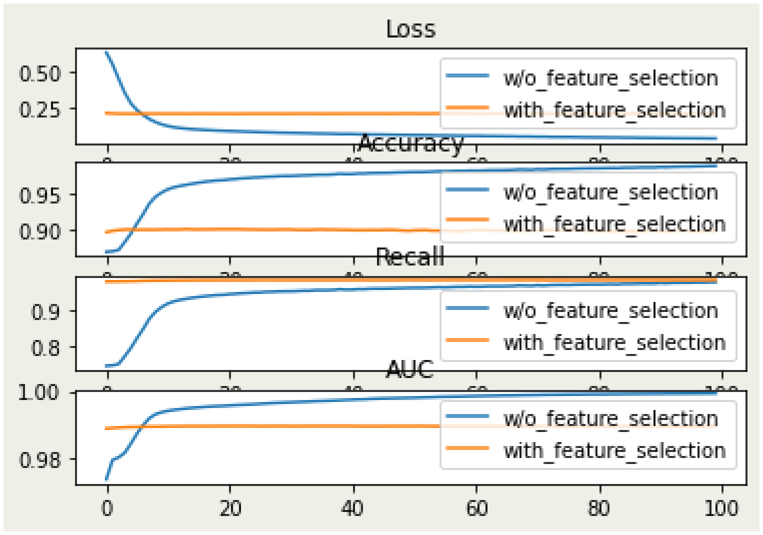


Figure 8 Comparing the performance of LSTM-Attention with and without feature selection during training

	Accuracy	Recall	AUC	Training time(s)
LSTM+Attention with feature selection	0.899	0.985	0.989	697.722
LSTM+Attention without feature selection	**0.987**	**0.979**	**0.999**	**686.847**

Table 6. Comparing the performance of LSTM +Attention with and without feature selection

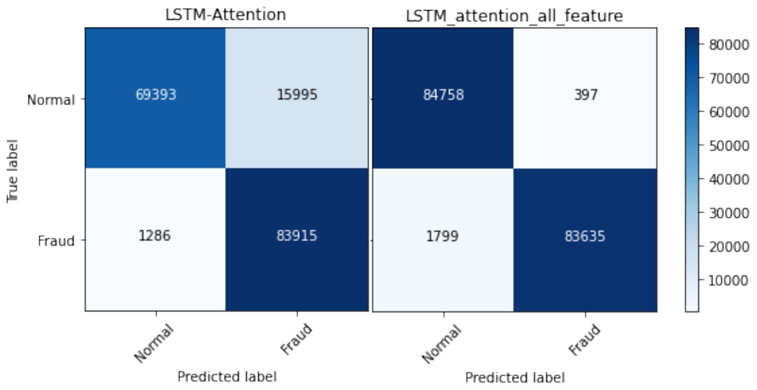


Figure 9 The confusion matrices of LSTM +Attention with and without feature selection

4.1.2.5 HYPERPARAMETER TUNING

- We used dropout equal to 0, 0.3, 0.5 when working with LSTM model and figured out that using no dropout, can help improve the recall value.
- We also changed the number of LSTM layers and tried 2, 3 and 4 layers and realised that increasing the number of layers can help with the model's performance.
- Visit supplementary part for more information.

4.1.3 Autoencoder

We also tried to study how autoencoders can perform in the credit card fraud detection problem. We tried to replicate "Credit Card Fraud Detection Using Autoencoder Model in Unbalanced Datasets" by Ali Shabi and to investigate what would happen if we changed the number of layers or the activation function type. We did not use feature selection; we used all 28 features (not including time and amount features). Their proposed architecture is named Case (3) in our report.

(1) Batch size = 128 - epoch:100 - ReLU activation function – learning rate: 1e-7
30 (input layer) – 14 – 7 – 7 – 30
(2) Batch size = 128 – epoch: 100 - ReLU activation function - learning rate: 1e-7
30 (input layer) – 22 – 15 – 10 – 15 – 22 – 30
(3) Batch size = 128 - epoch:100 – ReLU/tanh activation function – learning rate: 1e-7
30 (input layer) – 14 – 7 – 7 – 30
(4) Batch size = 32 - epoch:100 – ReLU/tanh activation function – learning rate: 1e-5

Model	AUC	Recall @ th = 5	Recall @ th = 3	Recall @th = 1
Case (1)	0.9293	0.6173	0.7826	0.8609
Case (2)	0.9270	0.6087	0.7826	0.8435
Case (3)	**0.9364**	0.6348	0.7913	0.8522
Case (4)	0.9351	0.6348	0.7913	0.8609

Table 3. Comparing the performance of autoencoders with different architectures

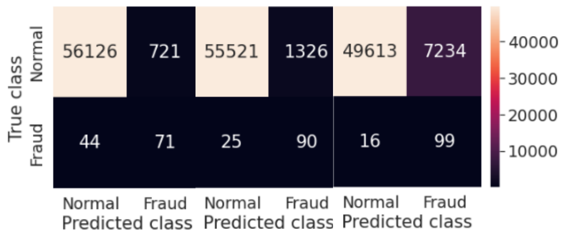


Figure 10 Case (1) – threshold = 5 and 3 and 1 (left to right)

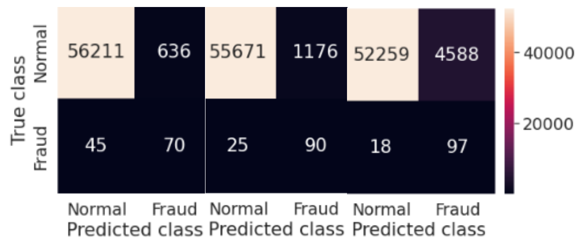


Figure 11 Case (2) – threshold = 5 and 3 and 1 (left to right)

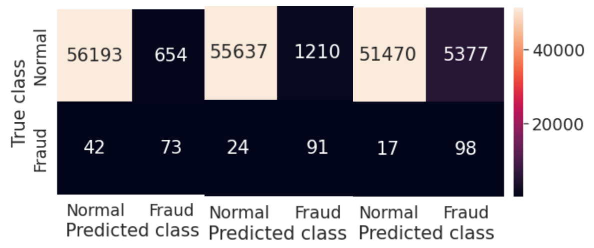


Figure 12 Case (3) – threshold = 5 and 3 and 1 (left to right)

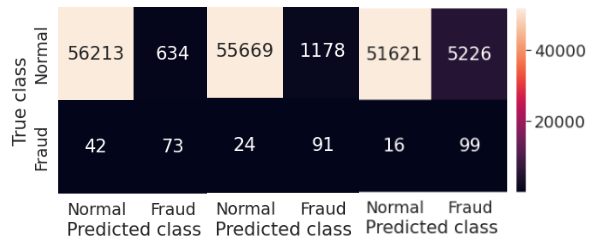


Figure 13 Case (4) – threshold = 5 and 3 and 1 (left to right)

To complete the article implementation, we also provided the following analysis for Case 3:

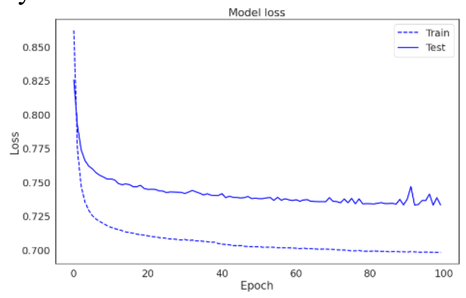


Figure 14 Reconstruction error vs. epochs

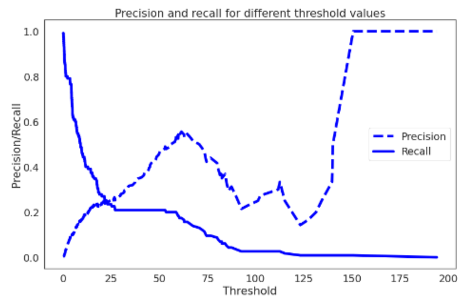


Figure 15 Precision and recall values with various values of the threshold

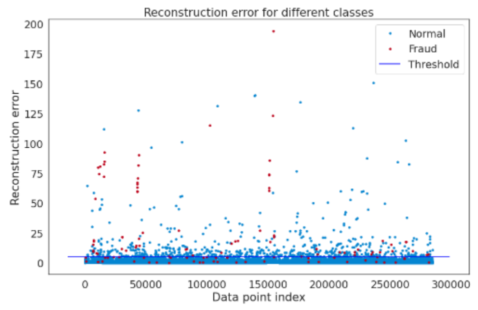


Figure 16 Data distribution in threshold 5

5 DISCUSSION

For the first article, we faced some problems when running the authors' code. According to what is stated in their paper, they have mentioned that they have done feature selection and ***feature extraction using UMAP*** and then have introduced the resulted components after applying UMAP to their proposed model. However, when we went through the code they provided, we could find the feature selection step. Then we noticed that those selected features are used directly as an input to the LSTM and LSTM + attention models, while we expected to see a UMAP step in between. We could not find this step in the code. Hence, we decided to implement this section by ourselves (part 4.1.1 is the model's results that we implemented by considering the explanations in the article, and the feature extraction step is involved). We also used the authors' code to replicate the results produced by inputting selected features directly into the models (part 4.1.2). The possible inconsistency between code and article is more to be investigated; we cannot and do not accuse the authors of scientific fraud. Still, we could not link the code and material in the article. So, for the first article, we have two implementations. First, we tried to reproduce things as mentioned in the article (part 4.1.1) and compare the LSTM and LSTM + attention models in terms of their performance. We also reproduced the results using the authors' code which seems not to include the feature extraction step (part 4.1.2); then, in this section, we tried to investigate the effect of data augmentation, feature selection and changing model's hyperparameter.

We were able to run all experiments within the second article and could see the reproducibility of their results; however, the authors have not published the codes. We also investigated other autoencoder architectures and saw how adding more encoder/decoder layers, changing activation functions within layers, and learning parameters can affect the model performance.

5.1 WHAT WAS EASY

- Talking about the first code, not considering the feature extraction part that we could not find within the code implementation, the good point was that the ipyn file of the code was included in GitHub. Hence, one could easily follow the code and the results it produces.

5.2 WHAT WAS DIFFICULT

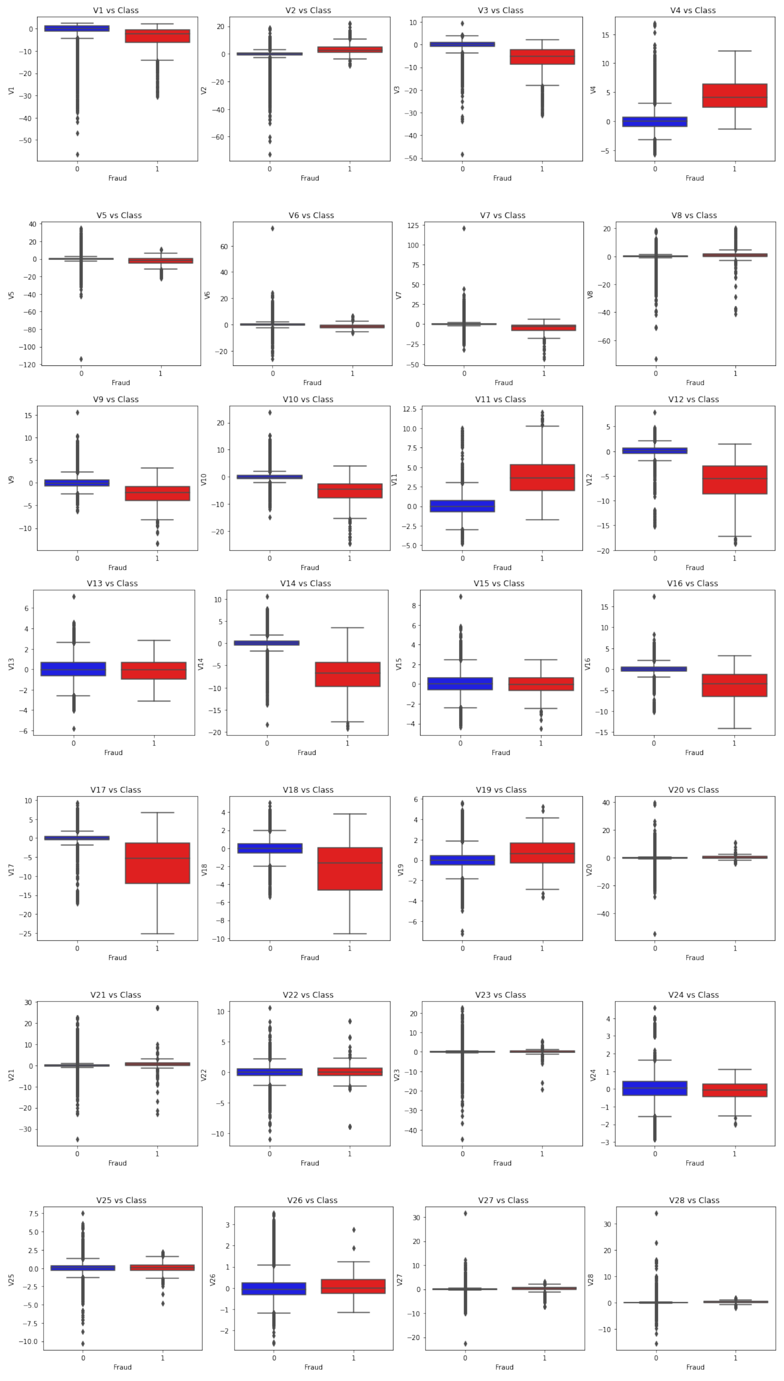
- For the first article, as mentioned before, the code seemed to miss the feature extraction step.
- No difficulties in terms of the second article implementation,

5.3 COMMUNICATION WITH ORIGINAL AUTHORS

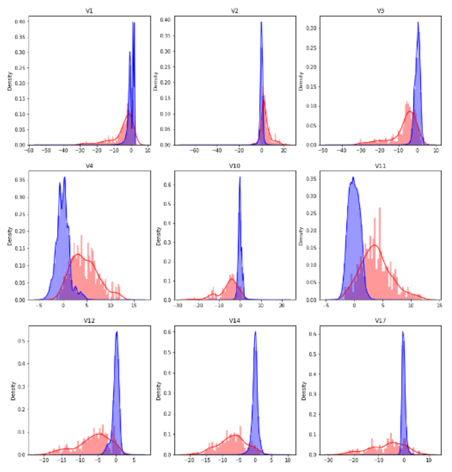
- For the first article that we mentioned, we left a comment on GitHub, and we emailed the corresponding author; however, by the time we were writing this report, we had not received any response.

6 Supplementary figures

6.1 Barplot of features per labels

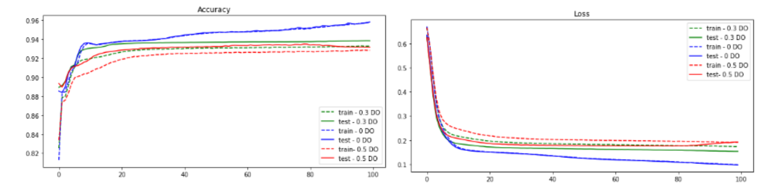


6.2 Distribution of Selected Features per label

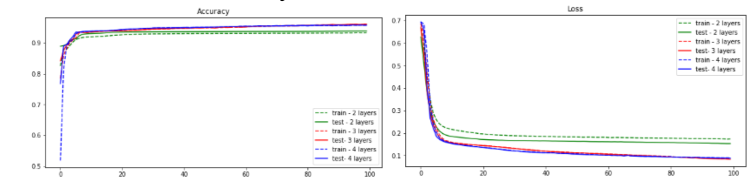


6.3 Hyperparameter tuning LSTM + SMOTE + no feature extraction

- Dropout



- Number of Layers



7 Refrences

1. Download the credit card fraud dataset Available: https://www.kaggle.com/mlg-ulb/creditcardfraud/data
2. Safdari, R., et al., *Applying data mining techniques to classify patients with suspected hepatitis C virus infection.* Intelligent Medicine, 2022.
3. Chawla, N.V., et al., *SMOTE: synthetic minority over-sampling technique.* Journal of artificial intelligence research, 2002. **16**: p. 321-357.
4. Benchaji I, Douzi S, El Ouahidi B, Jaafari J. Enhanced credit card fraud detection based on attention mechanism and LSTM deep model. Journal of Big Data. 2021 Dec;8(1):1-21.
5. Zou J, Zhang J, Jiang P. Credit card fraud detection using autoencoder neural network. arXiv preprint arXiv:1908.11553. 2019 Aug 30.
6. Zamini M, Montazer G. Credit card fraud detection using autoencoder based clustering. In2018 9th International Symposium on Telecommunications (IST) 2018 Dec 17 (pp. 486-491). IEEE.

8 Statement of contributions

Asa Borzabadi Farahani: LSTM (part 4.1.1) + autoencoders + its report

Gurucharan Marthi: report of part 3.2 + PCA implementation (not included in the report) + some data visualization

Yunhan Li: LSTM (part 4.2.1) + its report