ECE4179 - Neural Networks and Deep Learning

Semester 2 2021

Project Title: Comparing Fraud Detection Models

Dhruv Anand 30462452 dana0001@student.monash.edu

Submission Date: 24/10/2021

Downloaded Dataset csv and Trained Models: (can only access using Monash account)
https://drive.google.com/drive/folders/18vUJA2wUaw1bGaiHG6dTTmd6EDac6mb9?usp=sharing

Abstract

Credit Card Fraud Detection is one of the popular topics in the field of machine learning. There are many different methods that have been used to approach this problem in a more efficient way. The focus of this paper is to compare the performance of the autoencoder and a classical CNN classifier. The autoencoder is trained to only be able to reconstruct non fraudulent transactions and uses the difference between the original and reconstructed transaction to predict. Meanwhile, the CNN classifier has used a SMOTE balanced dataset. Through comparing these two models, the result has shown that CNN classifier has a better performance but it uses more parameters compared to autoencoders.

1.0 Introduction and Background

Fraud is when one party deceives another in order to receive some gain. In Australia alone, at least $211 million has already been lost as a result of fraud and scams, an 89% increase from the same period in 2020 [1]. As scams become more intricate, they get harder to detect and subsequently, harder to protect people from. Due to advancements in machine learning understanding, its widespread uses include fraud detection methods. The networks can be trained on a huge data set of legitimate and illegitimate transactions in order to learn the intricate characteristics which may determine the quality of the transaction and the nature of the person conducting it.

An autoencoder is a type of neural network which is given some data which it compresses into a bottleneck stage and then attempts to recreate the data in the same way that it received it as an input [2]. The value of this method is that when it compresses the input data it learns the features that are the most important in order to recreate the most important parts with as little data as possible. By using an autoencoder to learn to be able to represent non fraudulent transactions, when it sees a fraudulent transaction, it should not be able to recreate it, resulting in some error between the recreation and the input. If this error is greater than some threshold, the instance would be considered fraudulent.

Classifiers are a type of neural network model that determine the class of an input. In this case, whether a transaction is fraudulent or not. The model will be trained on a group of data points containing both fraud and non-fraud transactions and will train by using convolutions to produce a feature map and predict a result. Similar to the autoencoder, it will receive a one dimensional vector of features and produce a single output. The weights can then be analysed using a machine learning interpretation method, LIME (Local Interpretable Model-agnostic Explanations). This method is used to understand the model and explain its prediction by analysing which features it has prioritised.

By comparing the two models based on a number of metrics, more information will be able to be gathered about which method may be better in terms of determining the legitimacy of a transaction.

2.0 Related Works

Auto-encoder

In “Credit Card Fraud Detection Using Autoencoder Neural Network” (P Jiang et al, 2019) it is demonstrated that an auto encoder can remedy the issue of trying to use a minority class to train a classification neural network because of its strength in unsupervised learning of data sets. The investigation yielded an accuracy of 97.93% using oversampling and a denoising autoencoder. In this paper they have used the SMOTE Sampling method to oversample the data. They have also tried the without oversampling approach, but it is not as good as oversampling. However, we will use the without sampling approach idea as a baseline of our approach. We will not apply SMOTE sampling, instead we will try to make use of the imbalance data by doing the unsupervised method and training the model with just non fraudulent transactions.

Classifier

Dabakoglu [5] uses a classifier to predict the class of a transaction. They have used SMOTE sampling to balance the imbalance data and then use an ANN classifier to predict the class. Their research has shown that SMOTE is a useful algorithm to resolve the issue of over sampling. Although, after applying SMOTE, the ANN classifier has obtained a better result. However, the result is still having a low non fraudulent accuracy. Therefore, in this investigation we are intended to improve the accuracy by applying the SMOTE balanced data to a Convolution Neural Network (CNN) classifier.

3.0 Dataset

Kaggle Dataset Link: https://www.kaggle.com/mlg-ulb/creditcardfraud/version/3

The obtained dataset has 284,807 real transactions from European card holders in September 2013. Each instance contains 31 features including (1) the time of the transaction, (2) the amount, (3) class label (fraudulent = 1, not fraudulent = 0), and (4) 28 anonymised features labeled as V1 through V28. These 28 features have undergone principal component analysis (PCA). PCA involves standardising the variables, creating a covariance matrix, finding the eigenvalues and eigenvectors, creating a feature vector in order to determine the principal components and then using that data with the axes of the principal components [3]. As well as reducing the information lost and dimensionality, it also increases the interpretability and security of the users data that has been included in the whole set. 99.83% of the dataset were not fraudulent transactions while only 0.17% were fraudulent.

Data Pre-Processing

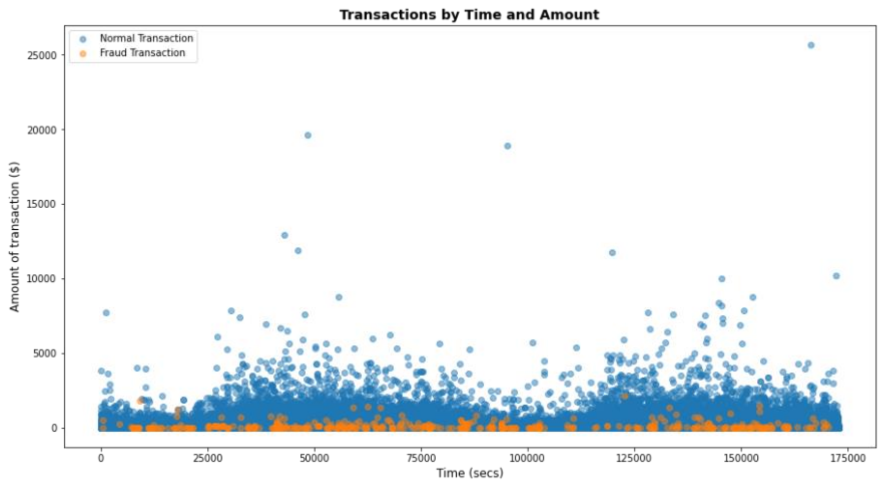


Figure 1: Amount vs Time for not fraudulent and fraudulent Transactions

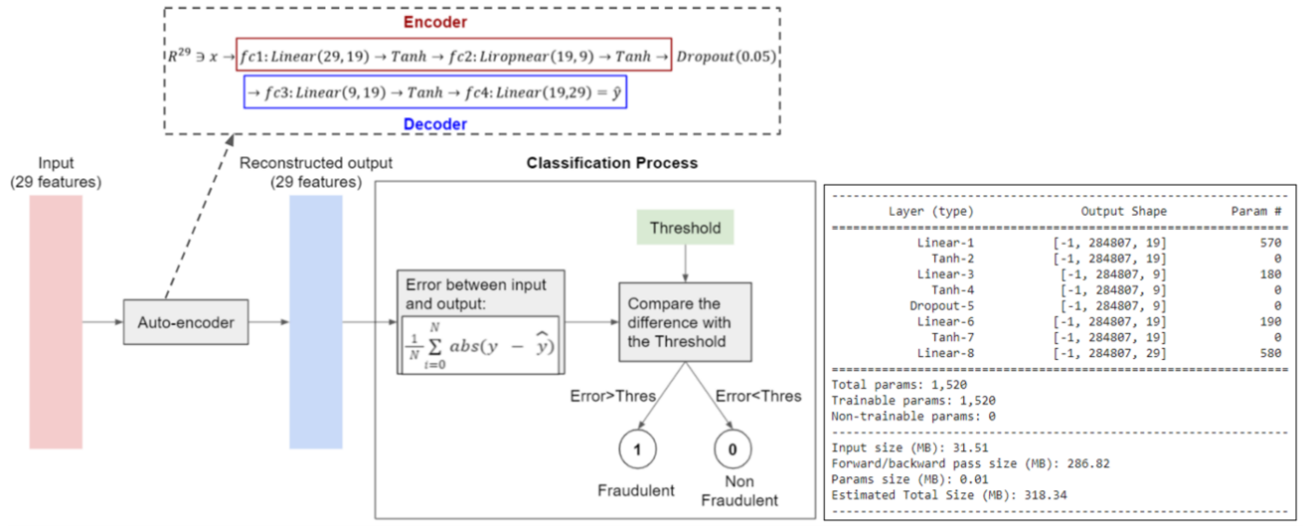
The scatter plot in Figure 1 has illustrated that the fraudulent transactions are occurring irrespective of the time. Therefore, this feature was removed from the dataset and not used in the analysis. Furthermore, the plot has shown that this dataset has included a few transactions which have a large transaction amount. In order to reduce the reliance on a few nodes and overfitting, the vast difference between the transaction amount is decreased by normalizing the transaction amount and also the PCA elements of this dataset.

For the **autoencoder**,non fraudulent transactions in the dataset is splitted into 70%, 10% and 20% to form the train, validation and test set respectively. The train and validation set have contained only not fraudulent transactions because the autoencoder is only trained to reconstruct not fraudulent transactions. This autoencoder uses unsupervised learning, thus the class label in the train and validation set are removed. Meanwhile, the test set included 20% of the non fraudulent transactions and all the fraudulent transactions. This is to ensure the test set has sufficient samples to study the performance of the model on detecting fraudulent transactions. Therefore, for evaluating purposes the class label of the test set is stored in the *"y_test"* vector. The normalizing process of the transaction amount and PCA elements is done outside of the model in this autoencoder approach.

For the **classifier**, the normalizing process is done by the Batch Normalization layer in the model. In this approach, this unbalanced dataset is balanced using the SMOTE(Synthetic Minority Oversampling Technique**)** sampling method due to the minority of data for fraud class. This can be achieved by simply duplicating examples from the minority class(fraud data) in the training dataset prior to fitting a model.This was implemented in order to minimise the issue of a minority class by creating more samples of fraudulent transactions. After sampling the data, it now consists of 284315 fraudulent and 284315 non-fraudulent transactions, a total of 568630 samples. This balanced data is splitted into 70% and 30% for the train and test set respectively. As the classifier has used supervised learning, thus the class label for both train and test set is stored in the "y_train" and "y_test" vectors.

4.0 Methods and Models

Autoencoder


Encoder
$R^{29} \ni x \rightarrow fc1: Linear(29,19) \rightarrow Tanh \rightarrow fc2: Liropnear(19,9) \rightarrow Tanh \rightarrow Dropout(0.05)$
$\rightarrow fc3: Linear(9,19) \rightarrow Tanh \rightarrow fc4: Linear(19,29) = \hat{y}$
Decoder
Input (29 features)
Auto-encoder
Reconstructed output (29 features)
Classification Process
Error between input and output:
$\frac{1}{N}\sum_{i=0}^{N} abs(y - \hat{y})$
Threshold
Compare the difference with the Threshold
Error>Thres
Error<Thres
1
0
Fraudulent
Non Fraudulent

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1          [-1, 284807, 19]             570
              Tanh-2          [-1, 284807, 19]               0
            Linear-3           [-1, 284807, 9]             180
              Tanh-4           [-1, 284807, 9]               0
           Dropout-5           [-1, 284807, 9]               0
            Linear-6          [-1, 284807, 19]             190
              Tanh-7          [-1, 284807, 19]               0
            Linear-8          [-1, 284807, 29]             580
================================================================
Total params: 1,520
Trainable params: 1,520
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 31.51
Forward/backward pass size (MB): 286.82
Params size (MB): 0.01
Estimated Total Size (MB): 318.34
----------------------------------------------------------------
```

Figure 2: Autoencoder Model and the parameters of this Model

The general idea of this approach is to train the autoencoder model to generate non fraudulent transactions by training it with just non-fraudulent samples. The ideal model is expected to have a good performance in reconstructing non fraudulent transactions. Thus, when the input of the model is a fraudulent transaction, the reconstructed output will have a large difference with the original fraudulent input because the model is not trained for fraudulent transactions. The error between the input and output will be compared with a threshold, if the error is larger than the threshold, the predicted class is fraudulent (predicted label =1). This process is illustrated in Figure 2.

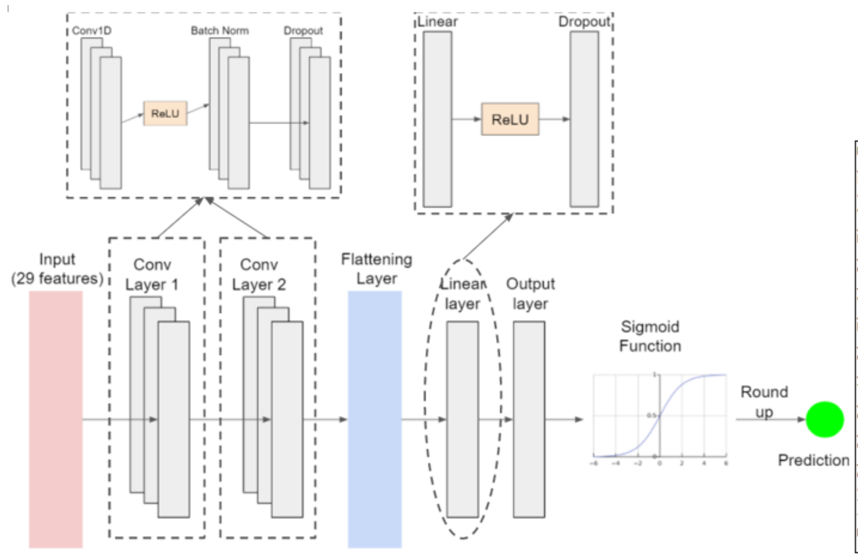
This autoencoder model consists of 4 fully connected layers, hyperbolic tangent (aka Tanh) activation function layers and a dropout layer. The fully connected layers reduce in nodes down to the bottleneck layers which are (19,9) and (9,19) respectively. The loss function used to train the network is mean squared error shown in Eq(1). Auto-encoder is not initially a classifier, thus softmax is not applicable in finding the prediction. MSE which oversees the squared difference between the true and predicted value which is the input and reconstructed output is perfect in determining the performance of the autoencoder. Furthermore, MSE is also the commonly used loss function for autoencoder models. The optimizer used is the Adam optimizer which normally has a lower learning rate than SGD, thus the learning rate used to train this model is 1e-4 with 100 epochs.

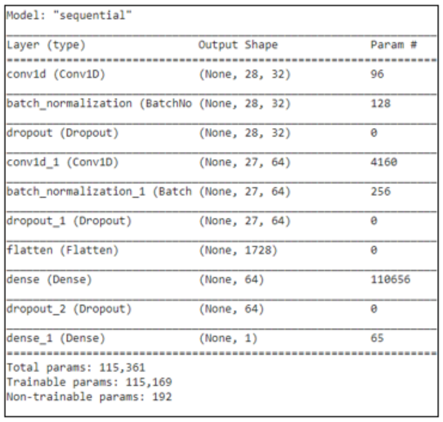
$$MSE: L(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N} (y - \hat{y})^2 \qquad \text{Eq(1)}$$

$$MAE: L(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N} abs(y - \hat{y}) \qquad \text{Eq(2)}$$

After the model outputs the reconstructed transaction, the difference between the original and reconstructed transactions is compared against the threshold. Threshold is computed using the validation set. First, **use the MEA function to determine the mean error for each sample in the validation set, then compute its mean of standard deviation. Threshold is simply just the addition of both mean and standard deviation.** MAE function is used to determine the difference input and output of validation and test set because of using the MSE function, the computed threshold is too low and it has decreases the non fraud accuracy. Threshold is not just straightly equal to the maximum value of the validation mean error because with this method the threshold is too high and decreases the fraud accuracy.

Convolutional Neural Network Classifier





```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 28, 32)            96
_________________________________________________________________
batch_normalization (BatchNo (None, 28, 32)            128
_________________________________________________________________
dropout (Dropout)            (None, 28, 32)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 27, 64)            4160
_________________________________________________________________
batch_normalization_1 (Batch (None, 27, 64)            256
_________________________________________________________________
dropout_1 (Dropout)          (None, 27, 64)            0
_________________________________________________________________
flatten (Flatten)            (None, 1728)              0
_________________________________________________________________
dense (Dense)                (None, 64)                110656
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 115,361
Trainable params: 115,169
Non-trainable params: 192
```

Figure 3: CNN Classifier and the parameters of this Model

A convolutional neural network has been implemented as the second method. This is because a simple CNN outperformed traditional machine learning algorithms (Support vector machines, Random Forest and Gradient Boosting Classifier). The model consists of two convolution layers and in each layer it consists of 1D convolution layer followed by batch normalisation layer and then dropout layer. The dataset is not normalized before passing into the model because the BatchNorm layer in the convolution layer is used to normalize the dataset. Following the convolution layers, the 3D data is flattened into 1D and passes through a fully connected layer. The output layer is stacked with this fc layer with a relu function in between. The output of the output layer is then passed through a sigmoid layer. Sigmoid: $f(x) = \frac{1}{1 + exp(-x)}$, using this function, the output of the last layer will be scale between 0 to 1. These values will then be rounded up to get an integer which then be used as the final prediction of the CNN classifier. This model used ‘Adam’ as its optimizer function and used the Binary cross entropy loss function. This loss function is used because the final output of the model is either 0 or 1 as there are only 2 classes in this dataset. By using this loss function, we can compare each of the predicted probabilities to its actual class stored in the y_test or y_train vector.

A simple CNN was developed to reduce the number of features and to easily identify differences between fraud and non-fraud transactions. Deep learning algorithms use deep architectures of multiple layers to extract features from raw data through a hierarchical progression of learned features using different layers from bottom navigating upwards without prior knowledge of any rule, which becomes more challenging when dealing with a huge amount of data.

5.0 Evaluation and Results

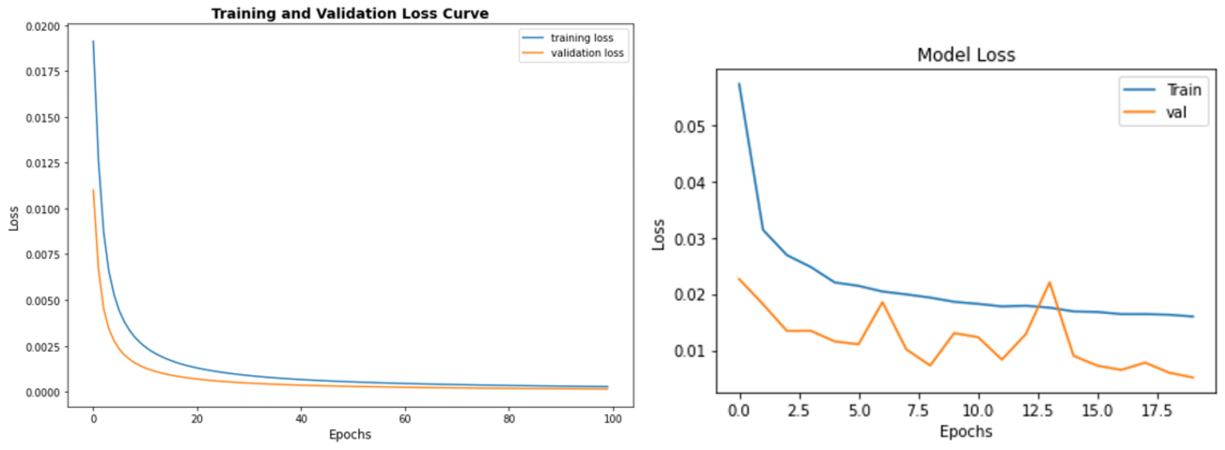


Figure 4: Loss curve of Autoencoder (left) and Classifier (right)

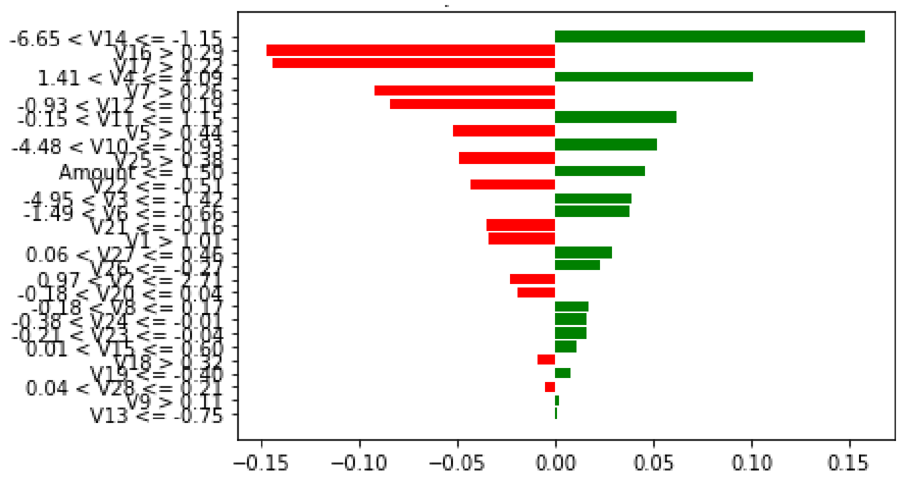


Figure 5: LIME explanation (right) of the Classifier is predicting a fraud instance

Table 1: Comparison Table

	Autoencoder	Classifier
Non Fraud Accuracy	94%	99%
Fraud Accuracy	87%	100%
Total Accuracy	94%	99%
Number of Parameters	1,520	115,361

Accuracy to Param Ratio	0.06	0.0009
Memory (KB)	10	300

5.1 Model Comparison

Upon training and evaluation of the two models, confusion matrices (in Appendix 1) have been constructed, which visually represent the binary classification results from the two methods. Clearly, in all categories of accuracy, the Classifier out performs the Autoencoder. The classifier identified every single fraudulent transaction with no false negatives and only 476 false positives. In this scenario, a false negative is much more harmful than a false positive as a false positive can be further verified but false negatives result in fraudulent transactions being undetected.

From seeing the loss vs epoch graph for both the models, the curve for Autoencoder converged much more smoothly than for Classifier which is very unstable and fluctuates due to very precise sensitivity but giving much less loss since it correctly predicted all the classes.

The sizes of the networks are also vastly different, the number of parameters in the autoencoder is 1,520 compared to 115,361, resulting in a difference of accuracy to parameter ratio 2 orders of magnitude smaller. The memory size of the CNN is 30 times as big as the autoencoder. This means that depending on the aims of the model, the autoencoder is more ideal for smaller memory applications where accuracy is not as vital. Compared to applications where size is not as much of an issue but accuracy is paramount.

Upon analysis of the classifier network, a LIME explanation of a fraud instance is shown in Figure 5. Green bars are features that suggest this instance is fraud, while the red bars are the features that suggest this is a non-fraud instance. We can see that the number of bar charts on both sides are almost the same (14 green bars and 13 red bars). The distribution of these bar charts has indicated that all the 29 features are used in the prediction and there are no useless, unnecessary input features. As there is one more green bar than red bars, thus the final prediction of this instance is fraud class which is correct. As illustrated in the chart, the features that give the most contribution to the prediction are V14, V16 and V17. Unfortunately due to the nature of the data, it is unknown what these features actually represent but this may be able to be determined to the detriment of the cardholder's information security. The autoencoder is not interpreted using the LIME algorithm because the autoencoder is not a typical classifier model and it doesn't have probability score and so the LIME is not able to interpret it as LIME depends on the probability scores.

6.0 Conclusion and Future Work

Advanced machine learning methods are becoming more necessary to protect against financial fraud. In order to better protect, better models must be implemented. The research conducted suggests that convolutional neural networks can achieve better accuracy in determining fraudulent transactions than autoencoder models. The CNN models, however, are much larger, diminishing their efficiency and number of use cases depending on the computing needs of certain applications. It was expected that the autoencoder would outperform the CNN due to the dataset being heavily skewed towards the non-fraud instances but due to the use of SMOTE sampling, this has enabled the CNN to achieve a high accuracy.

There is further investigation that could be done to increase the understanding of the best way to employ machine learning in fraud detection. Other models could be considered and compared like Generative Adversarial Networks to act in competition with a detector or the current models could be altered to provide a more comprehensive comparison. Another study could be done to combine several models to provide a larger structure that could achieve a greater degree of accuracy.

7.0 References

[1] Australian Competition & Consumer Commission. “Losses reported to Scamwatch exceed $211 million, phone scams exploding.” Scams. https://www.accc.gov.au/taxonomy/term/160 (Accessed Oct. 16, 2021)

[2] J. Jordan. “Introduction to autoencoders.” Jeremy Jordan. https://www.jeremyjordan.me/autoencoders/ (Accessed Oct. 16, 2021)

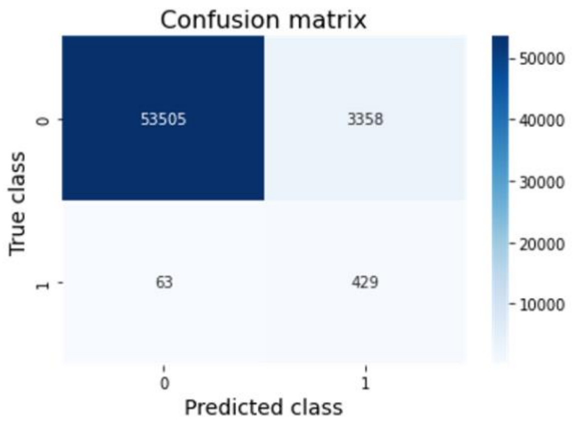
[3] Z. Jaadi. “A Step-by-Step Explanation of Principal Component Analysis (PCA).” builtin. https://builtin.com/data-science/step-step-explanation-principal-component-analysis (Accessed Oct. 19, 2021)

[4] P.Jiang, J.Zhang & J.Zou, “Credit Card Fraud Detection Using Autoencoder Neural Network,” Department of Electrical & Computer Engineer, University of Western Ontario, 2019. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1908/1908.11553.pdf

[5] C.Dabakoglu. “Fraud Detection - ANN and SMOTE Sampling with Python”. Medium. https://medium.com/@cdabakoglu/fraud-detection-ann-and-smote-sampling-with-python-c12f2f55c00a (Accessed Oct. 21, 2021)

8.0 Appendices

Appendix 1 - Confusion matrix of the Autoencoder and Classifier



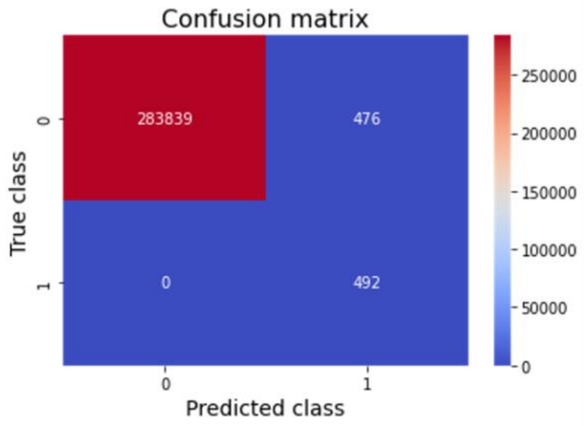


Figure 6: Confusion Matrix of Autoencoder (left) and Classifier (right)