Rotman School of Management
UNIVERSITY OF TORONTO

Rotman

Group Number: Group 9

Assignment Title: Final Project Report

Course Code: RSM 8521

Instructor Name: Brian Keng

In submitting this **group** work for grading, we confirm:

- That the work is original, and due credit is given to others where appropriate.
- That all members have contributed substantially and proportionally to each group assignment.
- That all members have sufficient familiarity with the entire contents of the group assignment so as to be able to sign off on them as original work.
- Acceptance and acknowledgement that assignments found to be plagiarized in any way will be subject to sanctions under the University's Code of Behaviour on Academic Matters.

Please **check the box and record your student number** below to indicate that you have read and abide by the statements above:

☒	1002365512	☒	1001755031
☒	1000841958	☒	1001104743
☒	1006118443	☐	

The following report outlines two deep learning methodologies that can be implemented for mobile payments fraud detection. As evident from the topic name, financial fraud is a direct threat to banks and credit card companies, in a report published by the US Federal Reserve in 2018: credit card merchants on average absorb 53% of their losses from fraudulent transactions[1]. In turn the research into fraud prediction has substantially increased over the years, especially with advances in modelling techniques and incorporation of deep learning methods. This directly ties into the future analytic career interests of the group members; through this analysis a better understanding of deep learning applications (and drawbacks) is achieved. However, one of the constraints is the data used for training the model; there exists a vast number of non-fraud transactions that heavily outsize the number of fraud transactions, causing the overall training dataset to be imbalanced and affecting the predictive accuracy of models.

In the proceeding sections two deep learning models will be discussed in terms of their applications for fraud detection: Feed Forward Neural Networks & Auto-Encoders. Similarly, numerous techniques that can be used to solve the data imbalance issue will also be discussed, those include: Synthetic Minority Over Sampling Technique (SMOTE) as well as Random Under sampling. It is important to note that the material presented in this report is sourced mainly from a research paper regarding *Fraud Detection using Machine Learning Techniques*, written by *Jurriaan Besenbruch*, from *Vrijie Universiteit Amsterdam*[2]. As a result, the models presented in this report are attempts of replicating those discussed in the paper, alongside experiments that were done in order to improve upon them and overall findings. Moreover, the dataset used is a PaySim simulation of mobile money transactions that are based on real transactions for a one-month period; sourced from Kaggle[3].

Description of Method

Data Imbalance

A notorious data integrity issue that exists when developing a fraud detection model is the severe imbalance in classes; evidently the majority of transactions that occur in everyday life are typically non-fraudulent. In the case of the PaySim dataset used for this report: from a total of 6,362,620 observations, there are only 8213 fraudulent transitions (less than 0.05%). The result is that the model does achieve around a 99% validation accuracy on prediction due to the large training sample available; however, the accuracy interpretation is misleading, since the model is biased towards predicting the majority class. Upon deployment this model will essentially be ineffective in the real world. Thus, two re-sampling techniques were experimented in order to try to maintain a balance between the two classes for prediction.

Synthetic Minority Over Sampling Technique (SMOTE) is a synthesizing method that creates new instances of the minority class; hence, it oversamples the minority class, while keeping the majority class constant. The algorithm creates synthetic observations by considering the k-

[1] https://www.federalreserve.gov/paymentsystems/files/debitfees_costs_2017.pdf
[2] https://science.vu.nl/en/Images/werkstukbesenbruch_tcm296-910176.pdf
[3] https://www.kaggle.com/ntnu-testimon/paysim1

nearest neighbors of the minority observations, and then creating new instances that follow a similar distribution; increasing the overall class size. An example is shown below in *Figure 1*, where the darker red circles represent the minority class observations, and the lighter smaller red circle represent the synthesized observations that lie within the surrounding neighbors of that class. The *imblearn.over_sampling*[4] package can be used in python to implement SMOTE, some the main parameters that can be adjusted are the number of neighbors that are considered when resampling and the ratio of resampled minority class observations to the majority class observations.

Some disadvantages also exist for SMOTE: Firstly, since the algorithm considered observational neighbors, it is not feasible for high-dimensional data where there are a lot of features, the curse of dimensionality comes into effect, where the distances between observations becomes negligible and significantly more data is required to maintain the same density as lower dimensional data. Additionally, SMOTE does not take into consideration that the neighboring observations may belong to a different class and as a result the synthesized observations may overlap classes, which increases the noise in the data[5].

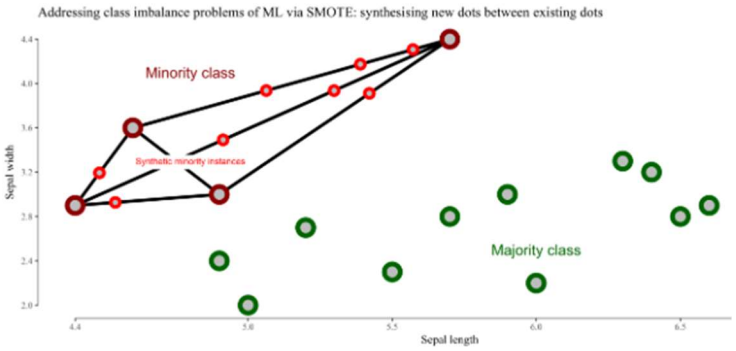


Figure 1 - SMOTE Algorithm Example

In contrast, another method to deal with imbalanced classes is to randomly eliminate observations of the majority class; Random Under Sampling Technique. This method is analogous to bootstrapping where re-sampling is occurring from the dataset. The benefits of this method are that both classes can eventually have the same number of observations, which theoretically should improve the training of the model. However, some drawbacks are that: Firstly, the bias in the data increases, since some valuable information that is evident in the overall data can be lost upon resampling. Secondly, the overall training sample size decreases significantly; in the PaySim case there are only around 8,000 fraud observations, so after under sampling both classes combined can have around 20,000 observations. This is not practical if the intention is to use a deep learning mode which requires a lot of training data. Moreover, the same python package as SMOTE can be implemented for random under sampling.

[4] https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html
[5] https://www.datacamp.com/community/tutorials/diving-deep-imbalanced-data

Feed Forward Networks

The first deep learning model that was used to predict fraudulent credit card transactions was a basic Feed Forward Neural Network; architecture shown below in *Figure 2* . The benefit of this model is that through the non-linear activation functions that are incorporated within the layers, the model was able to capture some abnormal spending trends such as variation in overall spending throughout weekdays or even hours of the day. However, the disadvantages of using this model exceeded any benefits. Firstly, due to the class imbalance problem the model would have almost a ~100% true positive rate, since there is no variation in the training data. In addition, upon under sampling there was not enough data for training and the model would perform poorly; the results are explained in the following section. The alternative deep learning technique that can be implemented is Autoencoders.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                768
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
=================================================================
Total params: 2,881
Trainable params: 2,881
Non-trainable params: 0
_________________________________________________________________
```

Figure 2 - Architecture of ANN

Auto-Encoder Network

An indirect approach for predicting financial fraud is through the implementation of an Auto Encoder network. This differs from the Feed Forward Network since the objective is not to actually classify observations based on probability; rather utilize the features of an Auto Encoder to learn a specific distribution and then input the test features and assess the resulting error.

Based on the autoencoder example shown in *Figure 3*, the architecture can be split into an encoder (before the bottleneck) and decoder (after the bottleneck). Upon inputting only *Non-Fraud* observations, the encoder layers capture the distribution and features of the sample (encoding the data). This distribution for data is then stored in the bottleneck layer, and reconstructed (recoded) in order to regenerate the *Non-Fraud* data once again. The metric being tracked is the *Reconstruction Error*, which measures the difference/error between the decoded output and the input observations.

Moreover, upon tuning parameters in order to optimize the network (discussed in the following section), the objective is to find network hyperparameters that minimize the reconstruction error when training on the *Non-Fraud* observations. Hence, in order to classify new observation (*Fraud* observations), upon inputting a *Fraud* to the network, theoretically, it would have a different distribution or features than those stored in the bottleneck; thus, the network would replicate

the *Fraud* observation, but with a high reconstruction error. The *cut-off* value for reconstruction error, which is a parameter that controls the threshold for classification, is tuned in order to maximize the predictive accuracy of the model, whilst maintaining a low false positive rate; meeting the business objective of a prospective user of this network for Fraud Detection. The parameter tuning is further discussed in the proceeding section.

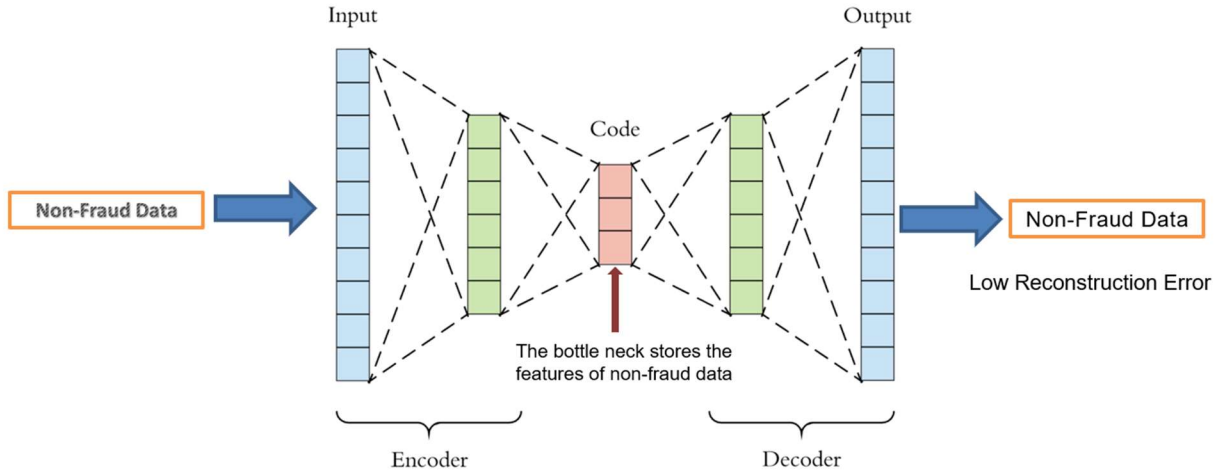


Figure 3 - Auto Encoder Architecture[6]

Experiments

Preprocessing

Before building the model, the data was pre-processed by changing categorical variable "type" to dummy variables, splitting dataset into train set (0.8) and test set (0.2) as well as rescaling training datasets via a StandardScaler and MinMaxScaler respectively. This aided in the learning of the model, since outlier observations were capped by the scaling and the categorical variables were separated into distinct classes.

The number of fraud and non-fraud data in the imbalanced dataset is *8213* and *6,362,620*, which negatively affected the learning ability of the model since the majority of observations only belonged to one class. In order to create a balanced dataset, the imbalanced dataset was divided into fraud and non-fraud datasets; *8213* non-fraud observations were randomly chosen from the overall dataset. Afterwards, a similar approach was taken for the SMOTE technique.

Feed-Forward Networks - Results & Analysis

Overall, six experiments were conducted to evaluate the effects of the dataset on the performance of a Feed Forward Neural Network, as well as the difference in classification between a traditional Feed Forward Network and Autoencoders in terms of fraud detection. Firstly, ***Standardized Imbalanced*** observations were used with the Feed Forward network. Since the output is binary, a sigmoid activation function is used in the output layer, and ReLU in the

[6] https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

hidden layers; Binary Cross entropy was implemented as the loss function. Surprisingly, the result of Binary Accruacy is around 99.78% and Validation Loss very close to 0; in contrast the False Positive Rate is 0.00%, as shown below in *Figure 4*. The results are indicative of a model that is memorizing the observations (essentially overfitting one class) due to the imbalance issue in the dataset; there are not enough *Fraud* class observations for the network to learn their features.

```
Percent of true positives: 30.27%
Percent of false negatives: 69.73%
Percent of true negatives: 100.00%
Percent of false positives: 0.00%
```

	Pred Fraud	Pred Non-Fraud
Actual Fraud	490	1129
Actual Non-Fraud	1	1270904

Figure 4 - ANN w/ Standardized Imbalanced Data

Secondly, ***Normalized Imbalanced*** observations were utilized in the same architectural network as before. The resulting Binary Accuracy was 99.85% and Validation Loss of 0; additionally, the confusion matrix can be seen below in *Figure 5*. Since the only change was the pre-processing technique used on the dataset, it can be concluded that a network that has been trained on Normalized data vs. standardized data has worse performance (even though they are both overfitting due to class imbalance); reflected through the increase in rate of false negatives of the model. Reasons could be that the current network architecture and parameters are more sensitive to input data being in the range of [0,1] (normalized) vs. outside that range (standardized).

```
Percent of true positives: 0.31%
Percent of false negatives: 99.69%
Percent of true negatives: 100.00%
Percent of false positives: 0.00%
```

	Pred Fraud	Pred Non-Fraud
Actual Fraud	5	1614
Actual Non-Fraud	1	1270904

Figure 5 - ANN w/ Normalized Imbalanced Data

Additionally, ***Under Sampling*** was implemented on the overall dataset in order to balance the fraud and non-fraud classes. The resulting Binary Accuracy dropped from previous methods to 50.30%, with a Validation Loss of 1.50. However, as seen in *Figure 6* the model is not overfitting as aggressively as previously; evident from the higher rates of False Positive predictions and True Negatives (w.r.t using imbalanced dataset). This shows that even though the model performance remains suboptimal and infeasible in real world application, by slightly altering the class observations there is a difference in training and predictive performance.

```
Percent of true positives: 98.72%
Percent of false negatives: 1.28%
Percent of true negatives: 1.45%
Percent of false positives: 98.55%
```

	Pred Fraud	Pred Non-Fraud
Actual Fraud	1615	21
Actual Non-Fraud	1626	24

Figure 6 - ANN w/ Under Sampling

Moreover, the dataset was modified through ***Over Sampling (SMOTE)*** and the Feed Forward network was reevaluated. The resulting Binary Accuracy is 89.19% with a Validation Loss of around 0.775. From the confusion matrix shown in *Figure 7* below, it can be seen that from an

academic perspective this model is performing the best, with 98.49% True Positive and only 2.00% False Positive. However, looking at the sample size, the False Positives actually represent around 25,444 transactions in the data. From a business perspective having that many False Positive might be more detrimental than maximizing the True Positive rate. Overall, ***SMOTE*** can be deemed to be the best method ,out of all proposed, in terms of modifying the dataset in order to improve model training.

```
Percent of true positives: 98.84%
Percent of false negatives: 1.16%
Percent of true negatives: 98.00%
Percent of false positives: 2.00%
```

	Pred Fraud	Pred Non-Fraud
Actual Fraud	1618	19
Actual Non-Fraud	25444	1245443

Figure 7 - ANN w/ Over Sampling

Autoencoders - Results & Analysis

As an alternative approach to deal with Fraud detection, the research paper investigated building an Autoencoder network. The main inituition behind building the Autoencoder network is driven by the scarcity of records classified as 'Fraud'. The imbalance in the training dataset is easily overcome because the network essentially trains to learn the underlying distribution of the majority class, 'Non-Fraud' in this case. *Figure 8* below represents the underlying architecture of the most optimal Autoencoder used in the reference paper. After multiple tuning and optimization, the hyperparameters of the replicated model is also shown in the same figure.

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_27 (Dense)             (None, 32)                384
_________________________________________________________________
dense_28 (Dense)             (None, 6)                 198
_________________________________________________________________
dense_29 (Dense)             (None, 6)                 42
_________________________________________________________________
dense_30 (Dense)             (None, 32)                224
_________________________________________________________________
dense_31 (Dense)             (None, 11)                363
=================================================================
Total params: 1,211
Trainable params: 1,211
Non-trainable params: 0
_________________________________________________________________
```

Hyperparameter	Value
Regularizer	L1
Learning Rate	1e-3
Number of Encoder Layers	2
Number of Decoder Layers	2
Hidden Layer Activation Function	ReLU
Output Layer Activation Function	Linear/Identity
Number of epochs	10
Number of Trainable parameters	1211
Optimizer	Adam
Metrics	Accuracy
Loss	Mean Squared Error

Figure 8 - Architecture & Parameters of Autoencoder

To evaluate whether the model is able to differentiate between a fraud and non-fraud dataset, the autoencoder network was evaluated on small subset of Fraud and Non-Fraud dataset. Below in *Figure 9* are the values of loss function, MSE, and accuracy of the model in the respective cases. It is evident that the reconstruction error from validating the model on Fraud and Non-Fraud data set is significantly different from each other. Hence, the model is performing well in identifying differences between Fraud and Non-Fraud data.

Data Set	Loss Function	Accuracy of Prediction
Validation Set – With Fraud	0.0043	0.9913
Validation Set – With Non-Fraud	1.105	0.1014

Figure 9 - Evaluation Results for Autoencoder

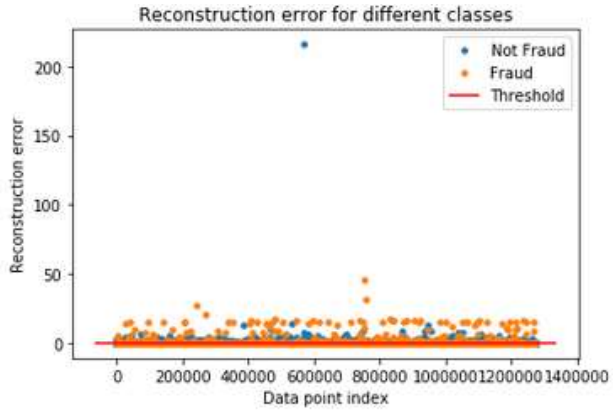
Extending the function of Autoencoder to make classifications for Fraud detection, it important to find the optimal threshold point for the reconstruction error. A transaction is considered fraud candidate according to the following rule[7],

$$x_k \rightarrow "normal" \; if \; \varepsilon_k \leq K$$
$$x_k \rightarrow "anomaly" \; if \; \varepsilon_k > K$$

where ε_k is the reconstruction error value for transaction x_k and K is a threshold.

Continuing with this idea, it is also important to take into account that in applications like Fraud detection in addition to improving the accuracy of the model, it is also significant to keep the False Positive rate as low as possible. This is because it is not desired to incorrectly classify legal or Non-Fraud transactions as Fraud. Having high False Positive rate may negatively impact the frequency of transactions or customer experience in case of using these models in financial institutions like banks.Taking into consideration these objectives, multiple iterations of True and False Postitive rate were calculated for different threshold values. The optimal **Threshold** value for **Reconstruction Error** of around ***0.009*** was achieved with **AUC of 0.88**.

With this threshold, the autoencoder was run on the test dataset. Based on the resconstruction error for each observation in the test data, it was either classified as Fraud or Non-fraud respective to the threshold value. **Accuracy** of **97.7%** and **False Positive** rate of about **2%** was obtained.The resultant confusion matrix as well as the classified observations, with respect to the threshold value is shown below in *Figure 10.*



	Pred Fraud	Pred Non-Fraud
Actual Fraud	599	1034
Actual Non-Fraud	27445	1243446

Figure 10 - Autoencoder Results

[7] https://towardsdatascience.com/anomaly-detection-with-autoencoder-b4cdce4866a6

Conclusion

As mentioned at the start of this report, the paper whose experiments in fraud detection using deep learning were replicated with similar results. While such models cannot currently be used for such applications in Canadian Financial Services, the results of the experiments reveal useful insights regarding the use of different models to handle the extreme data imbalance that can be found in the PaySim dataset and many other examples. To prevent excessive disruptions to customer experience caused by False Positive fraud detection, minimizing such occurrences can be considered more important for financial services firms than to maximize the True Positive detection rate without regard for the false-positive rate.

Deep Learning Experiments

The experiments reviewed in this report reveal that the most successful deep learning model replicated from the paper which inspired the report is surprisingly a simple Feed Forward ANN with Standardized Imbalanced Big Dataset. A possible reason why such a model performed best is that was trained on a distribution which was likely most representative of the true data generation process. The under-sampling and synthetic over-sampling variants of this model suffered from issues that make the resulting datasets contain distributions which vary widely from that of the test dataset. As a result, while the true-positive rate of the Feed Forward ANN with Standardized SMOTE Balanced Big Dataset was very high, the number of False Positives was also high. Although the False Positive rate was low in terms of percentage, the results of the predictions on the test dataset would have resulted in in more than 15 False Positives for every fraud transaction successfully detected. The implications on customer satisfaction that so many False Positive detections could cause makes this model practically inferior to the Feed Forward ANN with Standardized Imbalanced Big Dataset from a business KPIs perspective.

Other Approaches

The paper which inspired this report concluded that deep learning techniques tend to underperform the supervised statistical learning techniques, which is consistent with the findings of this report. The paper indicated that “condensed nearest neighbors” algorithm performed the best from among the algorithms it examines. Therefore, supervised statistical learning models are likely more appropriate for financial fraud detection in practice. Additionally, further modifications to the deep learning models which this report discussed could be experimented with, such as adding network graph data based on the origination and destination variables or using a different target balance when using SMOTE (rather than the default 50%-50% balance).

References

1. https://towardsdatascience.com/overcoming-challenges-when-designing-a-fraud-detection-system-29e33bcb43d2
2. http://rikunert.com/SMOTE_explained
3. http://www.dataminingapps.com/2016/11/what-is-smote-in-an-imbalanced-class-setting-e-g-fraud-detection/
4. https://www.datacamp.com/community/tutorials/diving-deep-imbalanced-data
5. https://imbalanced-learn.readthedocs.io/en/stable/under_sampling.html
6. https://s3.amazonaws.com/academia.edu.documents/32182007/Evaluation_Measures_for_Models_Assessment_over_Imbalanced_Data_Sets.pdf
7. https://science.vu.nl/en/Images/werkstukbesenbruch_tcm296-910176.pdf