Empirical Comparison of Deep Learning Approaches to Solve Credit Card Fraud

Rasmus Kær Jørgensen, Fiammetta Caccavale

Technical University of Denmark

ABSTRACT

Due to the advancement in computing technologies and e-commerce, the risk of fraud is dramatically increasing. The aim of the study is to provide a contribution to this issue. We present an empirical comparison of different Deep Learning approaches and techniques to detect fraud. We will assess their performance and try to infer what might have made some approaches successful and others less. Finally, we will quantify and illustrate the effects of the trade-off between True Positive Rate (TPR) and False Positive Rate (FPR) using Receiver Operating Characteristic (ROC) and confusion matrix, discussing the results.

Index Terms— Deep Learning, Autoencoder, Fraud Detection, Re-sampling Techniques, FNN

1. INTRODUCTION

The risk of fraud for financial services companies and their customers have dramatically increased in the last years by advancements in computing technologies and the expanding use of e-commerce platforms. The failure of identifying and prevent fraud is costing the industry billions of dollars per year, as estimated in a recent Nilson Report [1]. It is estimated that loss due to credit card fraud would top $27.69 billion in 2017, with an increase of 11% over the previous year [2].

Financial companies are now trying to find a solution to better protect themselves. Traditional approaches to detect credit card fraud are rule-based, employing a set of logic statements to query transactions and flag suspicious activity for human review. The disadvantage of these methods is that the rules can be so generalized that they cause millions of legitimate transactions to be turned away or declined. There is evidence that this approach is not as effective at detecting and exposing new fraudulent techniques in real-time [2]. Many AI techniques, including deep learning, are becoming increasingly popular and effective to solve this issue. The advantage of using deep learning algorithms is that they can quickly learn and analyze vast quantities of data and uncover anomalies, which are patterns in data that do not conform to a well-defined notion of normal behavior [3], or suspicious patterns in transactions [2]. Defining what anomalies are is relevant in our case because fraud detection is actually a kind of anomaly detection.

Since the 19th century, the statistical community has been studying how to detect anomalies and outliers in data [4]. Many researchers wrote extensive surveys on anomaly detection techniques, such as (Hodge and Austin, 2004) [5], who provide an overview of the techniques developed in machine learning and statistical domains. A review of novelty detection techniques [3] using neural network has been presented in Markou and Singh [6-7]. Regarding outliers, a considerable amount of research has been done in statistics and has been reviewed in books, such as (Rousseeuw and Leroy 1987; Barnett and Lewis 1994; Hawkins 1980) [8-10] as well as other articles [11-12].

In its survey on anomaly detection, Chandola [3] explains which aspects make this apparently simple approach very challenging. These aspects are related to the difficulty of defining normal region, and consequently discerning what are normal instances from the ones that are anomalous. These two groups are often not well separated, thus an anomalous observation which lies close to the boundary can actually be normal, and vice-versa. In many domains, normal behavior keeps evolving, therefore the notion of normal behavior might be outdated and not sufficiently representative of the future. Finally, another issue is that the data contains noise, which tends to be similar to actual anomalies, making it difficult to distinguish and remove [3].

2. AVAILABILITY OF RESOURCES

All the models are available on the GitHub account [13]. The notebook can be viewed online here: Notebook [14].

The code is commented and all the details about the implementations are explained carefully. We also further explain the choices regarding our models, we discuss the plots and present the steps of used techniques, as SMOTE and T-SNE.

3. DATA EXPLORATION

We will use the "**Credit Card Fraud Detection**" dataset, which contains transactions made by European cardholders in September 2013. The dataset presents 492 frauds out of 284807 transactions and it is highly unbalanced, where the positive class (frauds) account for 0.172% of all transactions" [15][1].

[1] See Appendix for the data visualization.

4. METRICS

Usually, the performance of machine learning algorithms or neural networks is evaluated using predictive accuracy. A problem might arise when the data is imbalanced: in our example, a basic classifier could predict the correct outcome with an accuracy of 99% simply by using a default strategy of guessing only the majority class.

However, the nature of our investigation (predicting fraudulent transactions), requires a fairly high rate of correct detection of the minority class. Therefore, simple predictive accuracy is not the optimal strategy in such a case [16].

The metric used to evaluate our models is the ROC curve. The increase of the ROC graphs in machine learning in the recent years is partly due to the realization that simple classification accuracy is often not the optimal metric for measuring performance [17-18]. Moreover, this metric is particularly useful for datasets with skewed class distribution and unequal classification error costs [19].

A ROC curve is used to depict the trade-off between hit rates (also called true positive rate and recall) and false alarm rates (false positive rate) of classifiers [20-21]. AUC is the area under the curve. A poor classifier has a AUC of around 0.5, which is random guessing, while the closer to 1 is the AUC, the better is the classifier.

To have a more objective evaluation of our networks, the results obtained will be compared to the best performing algorithms that solved the "Credit Card Fraud Detection" problem, which is an year old Kaggle competition [15] that has gained wide attention and interest. The problem is to classify transactions and discern fraudulent from genuine transactions in a highly imbalanced dataset (0.17% fraudulent vs 99.83% genuine). We compared our models only to those who use ROC AUC and not other metrics.

5. METHOD

We propose an empirical comparison of different deep learning methods, both supervised and semi-supervised, to solve the credit card fraud problem.

We implement different approaches: first, we will apply logistic regression and use the results obtained as baseline to assess the other models. Afterwards, we will perturb the data with different re-sampling techniques, since the dataset is highly imbalanced and many investigations show that this is a valuable method to handle the problem. We will feed the re-sampled data to a simple feedforward network and evaluate its performance. The last two approaches are more complex architectures: a deep neural network and an autoencoder. Finally we will evaluate and discuss the results.

The different architectures implemented are:

- **Simple Logistic Regression (baseline)**
- **FNN with re-sampling techniques**
- **Deep Neural Network**
- **Autoencoder**

5.1 Logistic Regression as baseline

We have chosen a logistic regression implementation from Sklearn [22] as the baseline method for analyzing and assessing the performance of our networks. Benchmarking against logistic regression provide a method for comparing the performance of the various approaches across different architectures.

We deliberately do not tune parameters, since we want to have a model as pure and simple as possible.

We obtained an AUC of 0.808. Although its performance seems quite high, this simple model seems not to be the optimal one. Tuning the parameter would have definitely increased the performance, but the aim of this classifier is not to be competitive, but to serve as baseline.

5.2 Supervised Approaches

"Techniques trained in supervised mode assume the availability of a training data set which has labeled instances for normal as well as anomaly class" (Chandola et al 2009) [3].

The approach we will take is to build a predictive model that will perform a binary classification between normal versus anomaly classes. The model will be validated by using unseen test data to test the models ability to generalize and determine which class each instance belongs to.

A major issue in supervised fraud detection is the imbalanced class distribution. Anomalous instances are far fewer compared to the normal instances in the training data. So, obtaining an accurate representation of the anomalous class is a challenge [3].

According to the literature, there are two good strategies to handle class imbalance: one is to assign distinct costs to training examples [23-24], the second is to use re-sampling techniques [20]. We decided to follow this second strategy.

We will try to handle the problem of having an imbalanced data distribution by re-sampling our original dataset in three ways:

- **undersample the majority class**.
- **oversample the minority class with SMOTE**.
- **combine SMOTE oversampling with Edited Nearest Neighbours**, which is found in the literature to be a very effective method to clean noisy samples[2] [16].

[2] Re-sampling techniques, the the difference between oversampling and undersampling and the details of the implementation are further explained in the notebook.

For each technique, we will feed the re-sampled data to a simple feedforward network and evaluate its performance.
Afterwards, we will implement a **deep neural network.**
We will compare the performance of the shallow models with perturbed data to the performance of a deep model, which uses the original dataset. The aim of this comparison is to investigate if a deep network will be able to capture more deep structures and patterns and hence, be able to make a fair discrimination between normal transactions and fraudulent transactions, provided no re-sampling or feature engineering of the input.

5.2.1 FNN with undersampling

The model implemented is a **Feedforward Neural Network** and, to increase the performance of our model, we perturbed the training data with random undersampling. Undersampling seeks to reduce the majority class instances in the training set [25]. The result of this operation is a significant reduction of the number of instances in the training set. A consequence and advantage of this method is also that the training time is greatly reduced. Moreover, there is a significant reduction in memory usage. A possible disadvantage of this approach is that, since we are randomly eliminating members of the majority class, it is possible that valuable information will be lost in the process, because the technique could accidentally delete data points that are useful to build an accurate model. Particularly, important information about the decision boundary between the two classes could be lost [26].
Visualization of the data after undersampling:

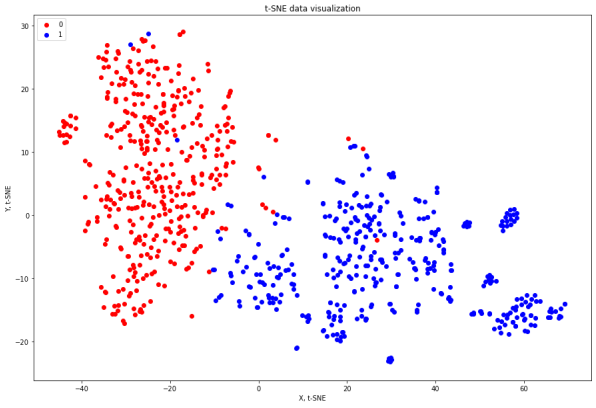


Figure 1: Visualization of the data after undersampling.

We can see that after undersampling the majority class, we obtained an equally distributed dataset. In fact, after undersampling we obtained a training set consisting of 401 normal transactions and 401 fraudulent transactions.

5.2.2 FNN with SMOTE

We implement a **Feedforward Neural Network** and, to increase the performance of our model, we manipulate the training data with SMOTE oversampling technique.
SMOTE (Synthetic Minority Over-sampling TEchnique) [16] is an oversampling technique in which the minority class is oversampled by creating 'synthetic' examples. It aims to enrich the minority class boundaries by introducing artificial examples in the minority class rather than replicating already existing examples to avoid the problem of overfitting [27].
Visualization of perturbed data:

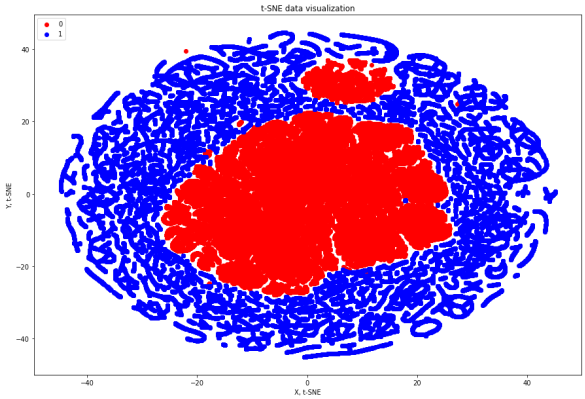


Figure 2: Visualization of the data after oversampling with SMOTE.

After applying oversampling with SMOTE, we can see that the ratio of normal and fraudulent transactions is rebalanced from an imbalanced to an equal class distribution. We obtained a training set consisting of 227444 normal transactions and 227444 fraudulent transactions.

5.2.3 FNN with a combination of oversampling with SMOTE and undersampling

Oversampling with SMOTE is, according to many researchers [27] a good way to handle the problem of imbalanced data. On the other hand, the literature [28-29] suggests that this method also introduces noisy samples (e.g. when the different classes cannot be well separated) by interpolating new points between marginal outliers and inliers. This issue can be solved by applying an undersampling algorithm to clean the noisy samples. Tomek links and Edited Nearest-Neighbours are two methods which have been used to obtain a cleaner space.
The implementations are both available at imbalance-learn library.
Generally, SMOTEENN [30] cleans more noisy data than SMOTETomek [31]. This is why we choose to use SMOTEENN in our study to re-sample the data. .

Undersampling of the majority class in ENN is done by removing points whose class label differs from a majority of its k-nearest neighbors [32].
The re-sampled data was used to train a **Feedforward Neural Network**.

5.2.4 Deep Neural Network

According to the Universal approximation theorem by Cybenko (1989) [34], a feedforward neural network with 1 hidden layer can approximate any function, provided the network is given sufficient hidden units. Thus, should be able to learn any 'learnable' fraud classification task.
With the implementation of this model, a **Deep Neural Network** with 3 hidden layers (1st hidden layer of 100 units, 2nd hidden layer of 70 units and 3rd hidden layer of 30 units) we wanted to explore if a more complex model can learn deeper structural aspect of the data, and have a high performance, even when the dataset is highly imbalanced and we do not perturb the data with re-sampling techniques.

5.3 Semi-supervised Approaches

"Techniques that operate in a semisupervised mode, assume that the training data has labeled instances for only the normal class" (Chandola et al 2009) [3].
This approach does not need labels for the anomaly class. Fraud scenarios can be difficult and challenging to model.

5.3.1 Autoencoder

Previously, we described what an anomaly is and we discussed why anomaly detection might be challenging. A common approach to detect anomalies is to define a "group of normal instances" where the normal non-fraudulent transactions lies and classify anything out of that "group" as an anomaly, given a certain measure. This is the method that we use in the implementation of the **Autoencoder**.
In fact, we present a model which is trained only with one-class non-fraudulent transactions. Concretely, we remove all fraud-cases from the training data such that the data only contains normal transactions. The aim is to learn a good compressed internal representation of the normal transactions, and thus reach a low reconstruction error for those cases. We will use this model to identify anomalies in the test data, which presents both normal and fraudulent transactions.
Our hypothesis is that, during the test phase, when the model is presented with a mix of fraudulent and non-fraudulent examples, it will be able to discriminate whether a transaction belongs to the fraudulent class or the non-fraudulent class based on the networks ability to errorlessly reconstruct the examples. The reconstruction error is expected to be high for the data who do not conform to the patterns of the non-fraudulent transactions learned from the training. Those with a high reconstruction error are the 'anomalies'. These anomalies can either be outliers or the fraudulent transaction that we are seeking to discriminate.

5.3.2 Application

Credit card fraud is an issue that many companies have to face, and therefore in the last years the interest in this area has grown tremendously. Not even big companies can afford to employ human supervision to detect all frauds cases.

When we have a large amount of data, which grows every day, it is not possible to afford to pay human resources to review each case, therefore we have to find a compromise between the amount of frauds detected and the number of cases that will be reviewed.
Practically, this choice can be done by setting a threshold: all the transactions whose reconstruction error is above this certain threshold will be classified as anomalous (potential frauds), and therefore will be passed on for human review, while the transactions with an error smaller than the set threshold will be considered normal transactions.
Setting the threshold value is more of a 'business decision' because it is indispensable to decide whether it is more valuable to minimize false positive rate (FPR), allowing the misclassification of some frauds (true negatives), or maximize the true positive rate (TPR), allowing the identification of many cases of misclassified normal transactions which have to be reviewed by human supervision (false positives).
The ROC curve is a very handy visualization of the problem.
In our case the optimal choice is to select a threshold which leads to the maximization of TPR and thus minimizes the number of cases in which a real instance of fraud was not flagged for review. Simultaneously, we can also reduce the FPR, meaning that we can spare resources that would have been used to cover the human revision of the cases in which non-fraudulent transactions were classified as frauds.

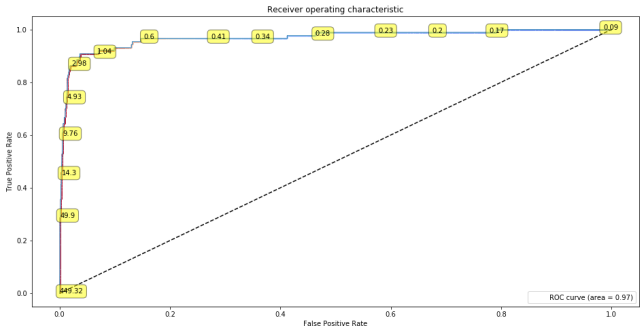


Figure 3: Visualization of the thresholds (autoencoder).

6. RESULTS AND DISCUSSION

We will now benchmark the results obtained against the best performing models from Kaggle and other project found online. We could only compare the models presented in this study with those models which used the ROC AUC metric.

ML approach	Name	ROC AUC
Autoencoder		**0.968**
Light GBM [35]	Niel Schneider	0.966
XGBoost [35]	Niel Schneider	0.965
RBM [36]	Weimin Wang	0.963
Autoencoder (Keras) [37]	Venelin Valkov	0.960
FNN w. undersampling		**0.953**
FNN w. combination SMOTE + ENN		**0.951**
GBM [35]	Niel Schneider	0.949
Autoencoder (Tensorflow) [36]	Weimin Wang	0.948
FNN w. SMOTE		**0.937**
Random Forest [38]	Gabriel Preda	0.930
Random Forest [39]	Christopher Taret	0.915
Logistic Regression + sampling [40]	Meena	0.912
Deep NN		**0.909**
Logistic Regression (Baseline)		**0.808**

Table 1: Table of Results. Table showing the performance of the models implemented in this study (grey rows) benchmarked against the best models from Kaggle and online.

The tables shows that the deep undercomplete autoencoder implemented in this study managed to achieve the highest AUC score. It outperformed the best performing model from Kaggle, which is a Light GBM by Niel Schneider, with AUC of 0.966 [35]. Our autoencoder also performed better than the other autoencoders presented in the table, which achieve, respectively, AUC of 0.960 and 0.948.

We can see that all the models presented in our study managed to outperform the logistic regression model used as baseline.
The table shows that the shallow models also achieves high performances. The FNN with undersampled data is the best performing model among the shallow networks.
The AUC of this model is 0.953, which is 0.145 higher than the baseline. Our assumption that reducing the instances of the majority class and having an equal distribution of instances could increase the performance of the model has been proved. Moreover, the results show that the model, even after the undersampling of the majority class, had enough examples to learn meaningful features about the data and be able to reconstruct the two classes with high performance.

Slightly lower was the performance of the FNN with SMOTE oversampling. We can see from the table that its AUC is 0.937. This approach effectively forces the decision region of the minority class to become more general. The hypothesis that rebalancing the dataset by creating synthetic observations from the minority class makes learning more effective and increase the generalization on the test data was verified. Furthermore, the FNN with SMOTE performs better than the simple logistic regression. The AUC of 0.937 is 0.13 higher than the baseline.
We managed to improve the performance of this model by cleaning the noise with Edited Nearest Neighbor. As suggested by the literature, a combination of oversampling with SMOTE and undersampling with Edited Nearest Neighbours produces good results and cleans the noise introduced by SMOTE. The hypothesis that SMOTEENN [30] is expected to provide a more clean decision surface [33] by removing “noisy examples”, and therefore increase the performance of our model, has been verified, both empirically and visually. This is shown by the improvement in the performance of the network and by the comparison of the plots of the re-sampled data.. It is also an indication that the strong oversampling introduced some noise.
The AUC of this model is 0.951, being 0.014 higher than the AUC of the FNN with SMOTE. It is outperformed by the feedforward network with undersampled data only by 0.002.

We observed that the shallow models managed to learn valuable features of the data, and their performance was high. The deep neural network achieves decent results, with an AUC of 0.909, outperforming the baseline by 0.101. Its performance is to be considered satisfactory, given the fact that the model was trained on the original imbalanced data.
The model seems to have learned valuable features, although not higher than the results achieved by shallow networks using

sampling techniques. We believe that a model like the deep neural network presented in this study could achieve great performances, but the amount of data available was scarce.
A disadvantage of this model is that is quite unstable and uneasy to implement, additionally tuning the parameters can be time consuming.
Since the aim of our study is partially to present a real-life problem and evaluate different methods that could be employed, we do not believe that this approach, given the amount of data we had, is the most optimal: it achieves satisfying results, but the instability of the model and the difficulty in tuning the parameters make it not the most recommended choice. Therefore we believe that better models might be more valuable to be used in this circumstances.

Both supervised and semi-supervised models achieved an high AUC, despite the imbalanced distribution of the classes.
The supervised models implemented, in combination with the re-sampled techniques applied (in the case of the FNNs) reveal to be well-suited models to solve the task.
On the other hand, the autoencoder shows to be an interesting and high-performance model to detect fraud in a credit card transactions dataset. It achieves the highest result, with an AUC of 0.968.

During the implementation of the models, we further explored the results obtained with the autoencoder and investigated which is considered the optimal threshold that simultaneously allows the maximization of the TPR and the minimization of the FPR.
We believe that the optimal threshold has to be found between 6.28 and 9.32[3]: this allow us to have a FPR close to 0 and a TPR higher than 60%. The selected threshold leads to an AUC of 80%, which we can consider very satisfying. Choosing a low threshold, such as 0.6 will allow us to catch the majority of fraud cases, but also more than 15000 non-fraudulent cases, which need to be reviewed by human supervision.
We believe that, regarding our practical example, the optimal choice of a threshold is: threshold = 9.69. With this threshold, in fact, we managed to correctly catch 53 fraudulent transactions, 56551 normal transactions and only 324 misclassified transactions will be reviewed by human supervision. We did misclassify 34 fraud cases, but we believe that it is a rather fair trade-off to misclassify few more examples of fraudulent transactions than have to review thousands more cases[4].

The following figure is a visualization of the reconstruction error for the two classes, with the chosen threshold.

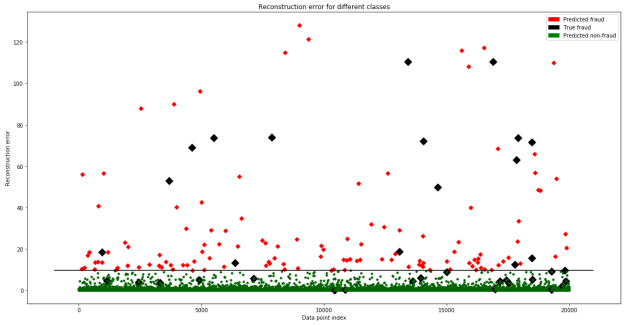


Figure 4: Visualization of the reconstruction error for 20000 data points. Black points are the TN, the green points are the TP and the red points are the FN.

What we can see from the visualization above is that all data points above **the threshold** are classified to be anomalies (predicted fraud) and data points below to be within the normal cases (predicted non-fraud). *Figure 3* shows the reconstruction error for the two classes: the black points are the actual frauds, the green ones are the data points classified as normal transactions and the red points are the false negatives which require human revision. The plot shows that there is no perfect classification into fraud and non-fraud cases, but the reconstruction error ratio is visibly higher for fraudulent transactions than for non-fraudulent. We consider every instance with reconstruction error > 9.69 to be anomaly/ outlier [1]. Moreover, the model allows us to retrieve all the predicted fraudulent cases for further investigation.

7. CONCLUSION

We believe that this study provides a contribution to the research in the field of anomaly detection and, more specifically, in fraud detection. We implemented and evaluated different methods, comparing supervised and semi-supervised learning. We showed that semi-supervised learning produces greater results in this case, but also re-sampled data, combined with shallow neural networks, can produce satisfying performances.

8. PERSPECTIVE

How could we improve and catch the more difficult cases?
to detect the few and more difficult instances of fraud cases as those with a low reconstruction error in the case with the autoencoder, we most likely need more data and additional features to learn from. The enrichment could be more sequential patterns to investigate such as behavioural purchasing habits, a time series such as spending patterns, or geographical information about the location of transaction and so on.

[3] See Appendix.
[4] See Appendix for the confusion Matrix.

9. REFERENCES

[1] The Nilson Report. Available at: https://www.nilsonreport.com/upload/content_promo/The_Nilson_Report_10-17-2016.pdf [Accessed 20/12/2017].

[2] Leveraging Deep Learning for Fraud Detection
By Pankaj Goyal, Vice President, AI Business & Data Center Strategy, Hewlett Packard Enterprise. Available at: https://www.hpcwire.com/solution_content/hpe/financial-services/leveraging-deep-learning-fraud-detection/ [Accessed 20/11/2017].

[3] Chandola, V., Banerjee, A., and Kumar, V. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, *41*(3), 15.

[4] Edgeworth, F. Y. 1887. On discordant observations. Philosophical Magazine 23, 5, 364–375. To Appear in ACM Computing Surveys, 09 2009.

[5] Hodge, V. and Austin, J. 2004. A survey of outlier detection methodologies. Artificial Intelligence Review 22, 2, 85–126.

[6] Markou, M. and Singh, S. 2003a. Novelty detection: a review-part 1: statistical approaches. Signal Processing 83, 12, 2481–2497.

[7] Markou, M. and Singh, S. 2003b. Novelty detection: a review-part 2: neural network based approaches. Signal Processing 83, 12, 2499–2521.

[8] Rousseeuw, P. J. and Leroy, A. M. 1987. Robust regression and outlier detection. John Wiley & Sons, Inc., New York, NY, USA.

[9] Barnett, V. and Lewis, T. 1994. Outliers in statistical data. John Wiley and sons.

[10] Hawkins, D. 1980. Identification of outliers. Monographs on Applied Probability and Statistics.

[11] Beckman, R. J. and Cook, R. D. 1983. Outlier...s. Technometrics 25, 2, 119–149

[12] Bakar, Z., Mohemad, R., Ahmad, A., and Deris, M. 2006. A comparative study for outlier detection techniques in data mining. Cybernetics and Intelligent Systems, 2006 IEEE Conference on, 1–6.

[13] GitHub account. Available at: https://github.com/FiammettaC/DeepLearning.

[14] Project Notebook. Available at: http://nbviewer.jupyter.org/gist/FiammettaC/719f464dae07ba3dfac4469b631a561d.

[15] Kaggle: Credit Card Fraud Detection. Available at: https://www.kaggle.com/dalpozz/creditcardfraud [Accessed 15/11/2017].

[16] Chawla et al. 2002. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research 16 (2002) 321–357.

[17] Provost, F., Fawcett, T. 1997. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In: Proc. Third Internat. Conf. on Knowledge Discovery and Data Mining (KDD-97). AAAI Press, Menlo Park, CA, pp. 43–48.

[18] Provost, F., Fawcett, T. 1998. Robust classification systems for imprecise environments. In: Proc. AAAI-98. AAAI Press, Menlo Park, CA, pp. 706–713.

[19] Fawcett, T. 2005. An introduction to ROC analysis Tom Fawcett Institute for the Study of Learning and Expertise, 2164 Staunton Court, Palo Alto, CA 94306, USA

[20] Egan, J.P. 1975. Signal detection theory and ROC analysis, Series in Cognition and Perception. Academic Press, New York.

[21] Swets, J.A., Dawes, R.M., Monahan, J. 2000. Better decisions through science. Scientific American 283, 82–87.

[22] Sklearn: Logistic Regression. Available at: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html [Accessed 15/11/2017].

[23] Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., and Brunk, C. 1994. Reducing Misclassification Costs. In Proceedings of the Eleventh International Conference on Machine Learning San Francisco, CA. Morgan Kauffmann.

[24] Domingos, P. 1999. Metacost: A General Method for Making Classifiers Cost-sensitive. In Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 155–164 San Diego, CA. ACM Press

[25] Kubat, M., Matwin, S. 1997. Addressing the Curse of Imbalanced Data Sets: One-Sided Sampling, in Proceedings of the Fourteenth International Conference on Machine Learning , pp. 179-186.

[26] Yun-chung Liu, A. 2004. The Effect of Oversampling and Undersampling on Classifying Imbalanced Text Datasets. Available at: https://pdfs.semanticscholar.org/cade/435c88610820f073a0fb61b73dff8f006760.pdf [Accessed 16/12/2017].

[27] Elhassan T., Aljurf M., Al-Mohanna F., and Shoukri M. 2016. Classification of Imbalance Data using Tomek Link (T-Link) Combined with Random Under-sampling (RUS) as a Data Reduction Method. Available at: http://datamining.imedpub.com/classification-of-imbalance-data-using-tomek-linktlink-combined-with-random-undersampling-rus-as-a-data-reduction-method.pdf [Accessed 16/12/2017].

[28] Imbalanced-learn: Combination of over- and under-sampling. Available at: http://contrib.scikit-learn.org/imbalanced-learn/stable/combine.html#combine [Accessed 15/11/2017].

[29] Imbalanced-learn: Comparison of the combination of over- and under-sampling algorithms. Available at: http://contrib.scikit-learn.org/imbalanced-learn/stable/auto_examples/combine/plot_comparison_combine.html#sphx-glr-auto-examples-combine-plot-comparison-combine-py [Accessed 15/11/2017].

[30] Imbalanced-learn: SMOTEEN. Available at: http://contrib.scikit-learn.org/imbalanced-learn/stable/generated/imblearn.combine.SMOTEENN.html#imblearn.combine.SMOTEENN

[31] Imbalanced-learn: SMOTETomek. Available at: http://contrib.scikit-learn.org/imbalanced-learn/stable/generated/imblearn.combine.SMOTETomek.html#imblearn.combine.SMOTETomek

[32] More, A. 2016. Survey of resampling techniques for improving classification performance in unbalanced datasets. Available at: https://arxiv.org/pdf/1608.06048.pdf [Accessed 15/11/2017].

[33] Handling imbalanced dataset in supervised learning using family of SMOTE algorithm. Available at: https://www.datasciencecentral.com/profiles/blogs/handling-imbalanced-data-sets-in-supervised-learning-using-family.

[34] Cybenko, G. 1989. Math. Control Signal Systems, 2: 303-314: Approximation by Superpositions of a Sigmoidal Function.

[35] Kaggle: GBM vs xgboost vs lightGBM by Neil Schneider, FS. Available at: https://www.kaggle.com/nschneider/gbm-vs-xgboost-vs-lightgbm [Accessed 5/12/2017].

[36] Kaggle: RBM and Deep Auto-encoder in TensorFlow to get AUC of 0.95 by Weimin Wang. Available at: https://www.kaggle.com/dalpozz/creditcardfraud/discussion/37626 [Accessed 15/11/2017].

[37] Valkov, Venelin. (2017). Credit Card Fraud Detection using Autoencoders in Keras — TensorFlow for Hackers (Part VII). Available at: https://medium.com/@curiousily/credit-card-fraud-detection-using-autoencoders-in-keras-tensorflow-for-hackers-part-vii-20e0c85301bd [Accessed 5/12/2017].

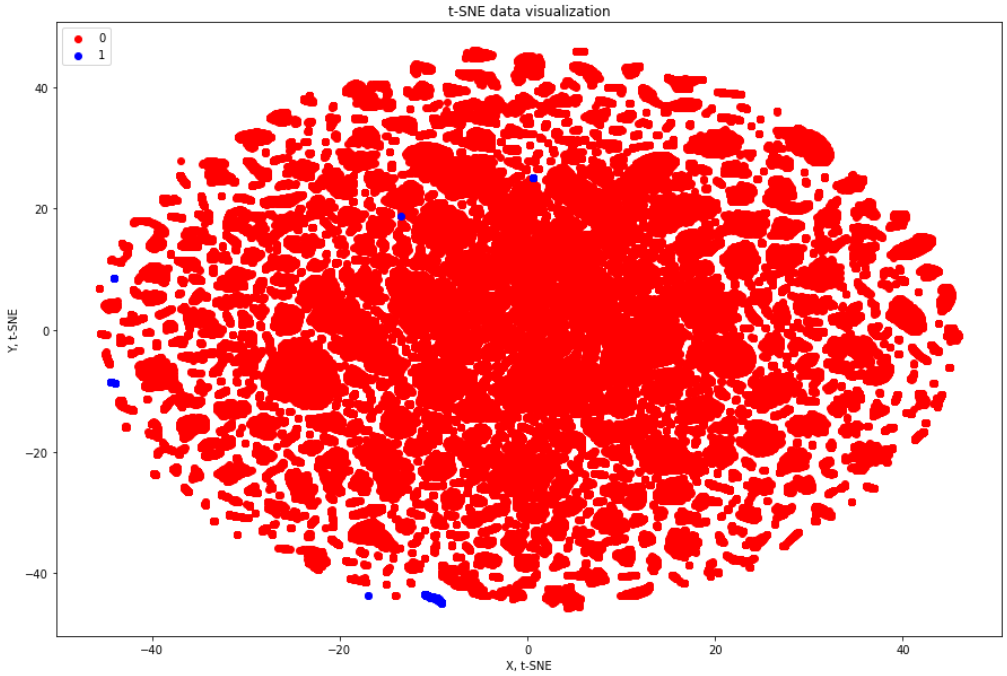
[38] Kaggle: Credit Card Fraud Detection with RF (AUC=0.93) by Gabriel Preda. Available at: https://www.kaggle.com/gpreda/credit-card-fraud-detection-with-rf-auc-0-93 [Accessed 5/12/2017].

[39] Kaggle: Fraud detection with SMOTE and RandomForest by Christopher Taret. Available at: https://www.kaggle.com/chtaret/fraud-detection-with-smote-and-randomforest [Accessed 5/12/2017].

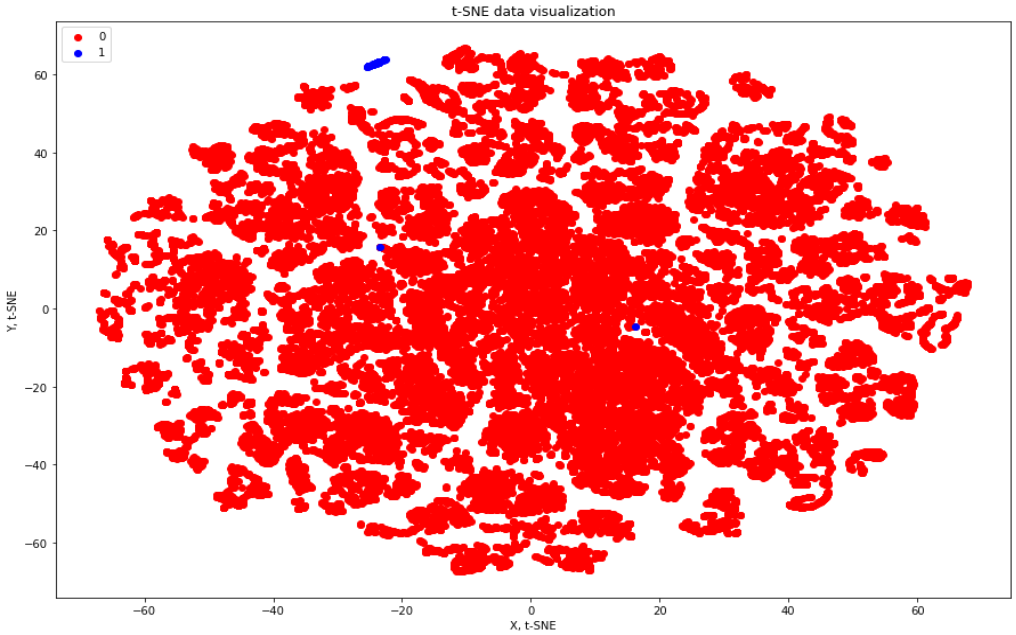
[40] Kaggle: Fraud detection by Meena. Available at: https://www.kaggle.com/meenaj/fraud-detection

10. APPENDIX

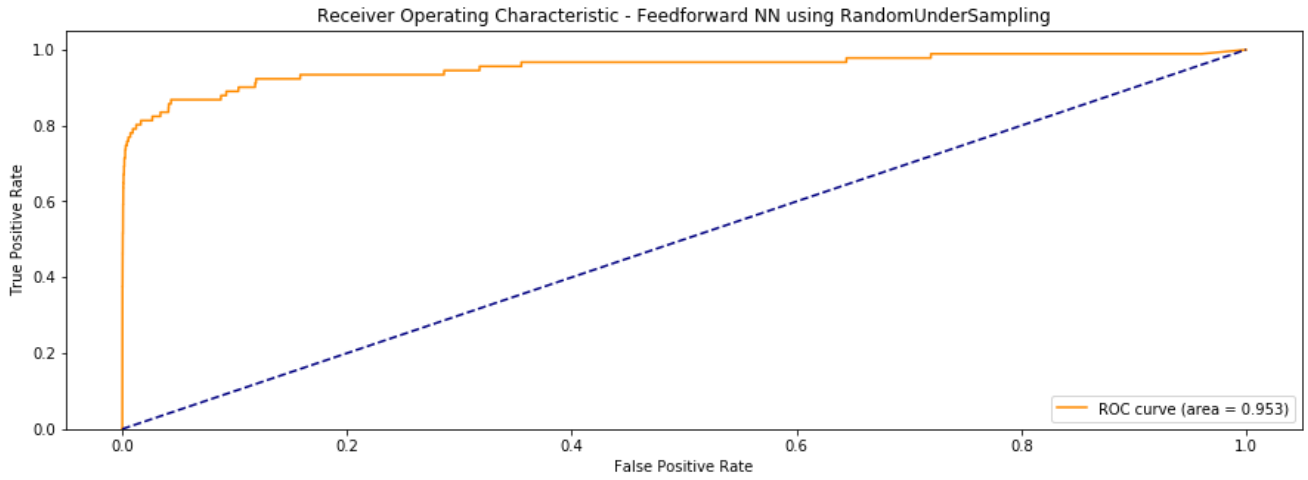
The appendix presents the plots of the ROC curve of the models and AUC scores. We could not insert all the plots in the report because we believed they would have been too small. The plot below is a visualization of the imbalanced distribution of the total dataset. The data is visualized using T-SNE. The red data points are the normal transactions, while the blue points are frauds.



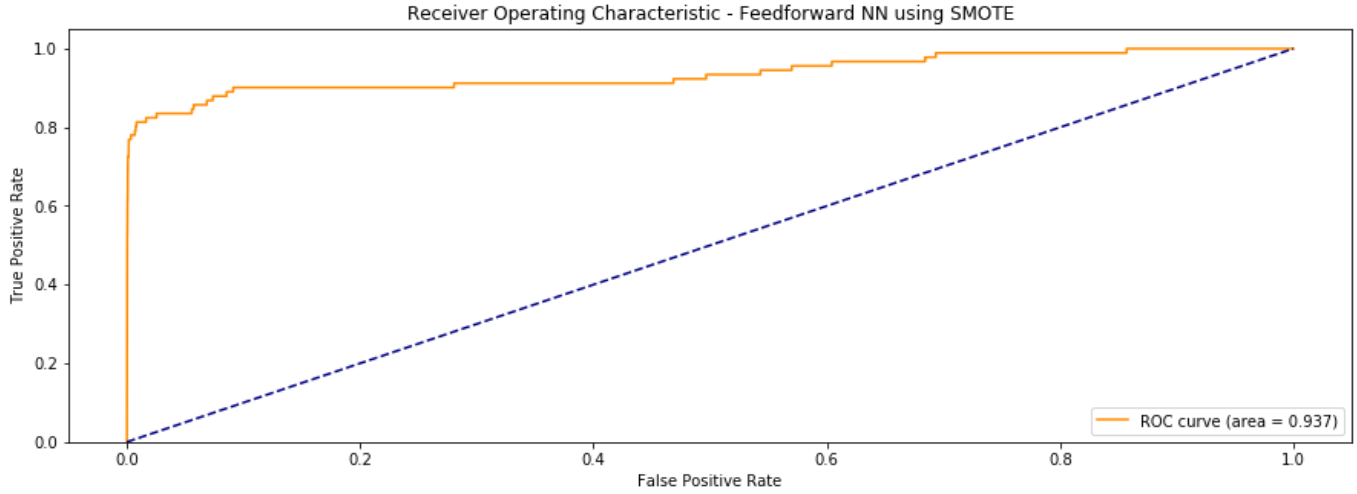
T-SNE visualization of test data:



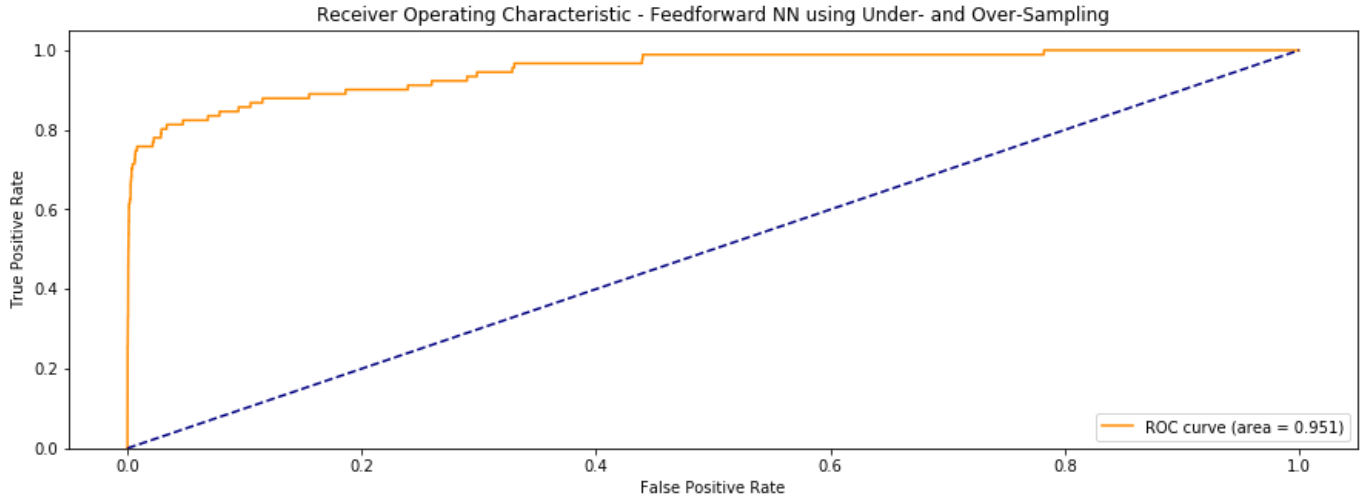
The following plot is the ROC curve of the FNN with undersampling:



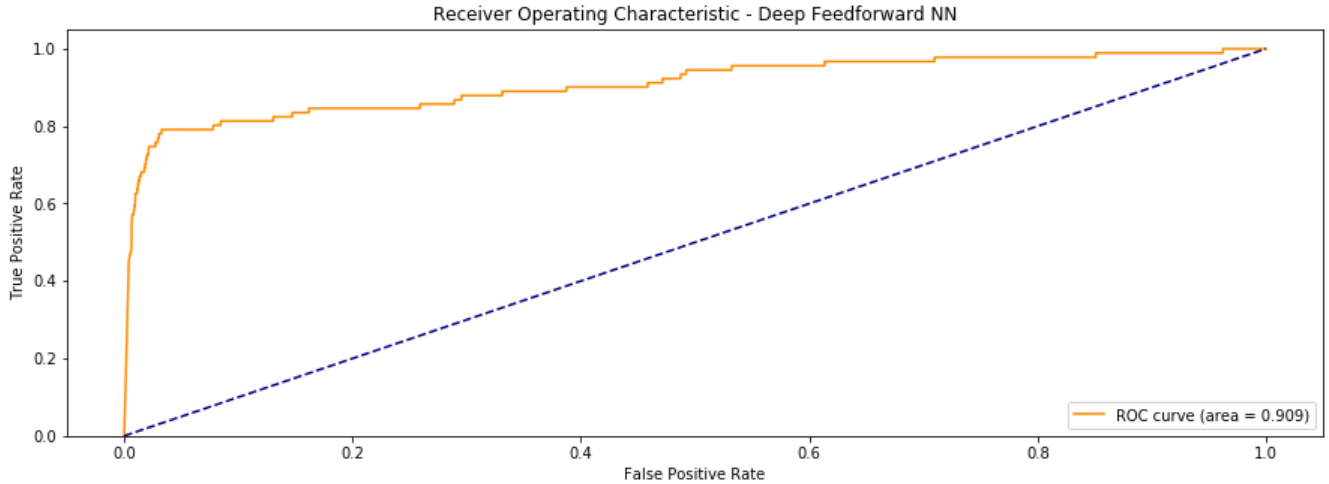
The following plot is the ROC curve of the FNN with oversampling with SMOTE:



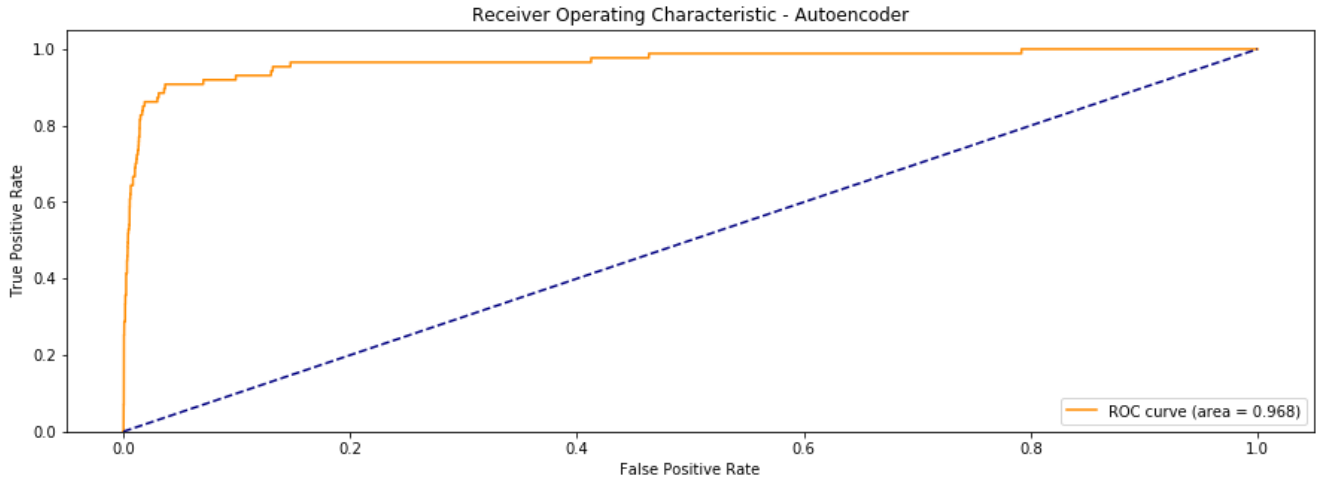
The following plot is the ROC curve of the FNN with oversampling with SMOTE and Edited Nearest Neighbours:



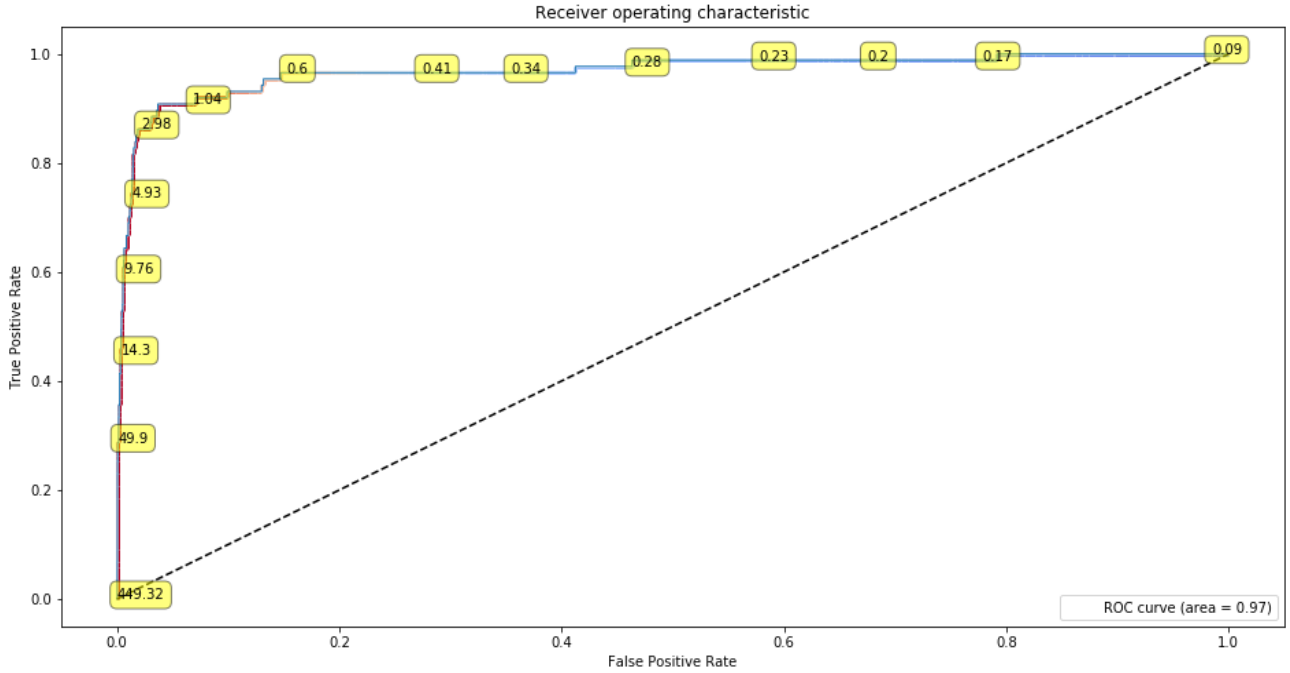
The following plot is the ROC curve of the DNN:



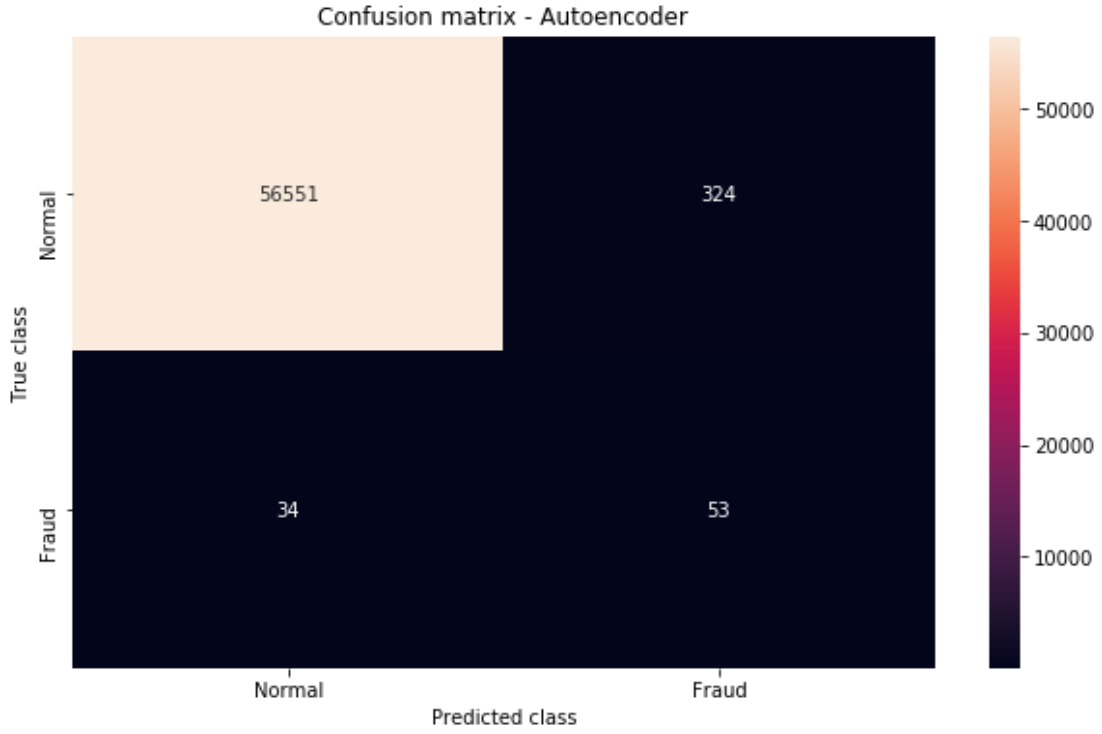
The following plot is the ROC curve of the Autoencoder:



The following plot is the ROC curve of the Autoencoder with all the possible threshold:



Confusion matrix with threshold = 9.69



The following plot is the reconstruction error of the two classes::