Deep Learning-Based Credit Card Fraud Transaction Detection

Sandeep Reddy Venna (101214885)[1*] and Jean-Gabriel Gaudreault (101252717)[1*]

[1*]School of Electrical Engineering and Computer Science, University of Ottawa, 75 Laurier, Ottawa, K1N6N5, Ontario, Canada.

*E-mails: svenn075@uottawa.ca; j.gaudreault@uottawa.ca;

April 12, 2022

Abstract

The number of customers who use credit cards has increased over the last decade. Credit cards are known for increasing the users' purchasing power and allowing them to meet their daily needs and much more. With the increased usage of credit cards, the number of credit card frauds has also increased dramatically. Credit card fraud is a serious criminal offense that costs consumers and financial organizations billions of dollars each year. As a result, financial institutions must be proactive about detecting and preventing these fraudulent activities. In this work, we explore the use of deep learning-based models for identifying these fraudulent transactions with acceptable performances. We use and compare machine learning techniques such as artificial neural networks, automated machine learning, and semi-supervised learning approaches and provide recommendations on which techniques could be used effectively for this type of task.

Keywords: Deep Learning, Fraud Detection, Credit Card Transactions, AutoML, Autoencoders

Table of Acronyms

Acronym	**Description**
ANN	Artificial Neural Network
AutoML	Automated Machine learning
CART	Classification And Regression Trees
DL	Deep Learning
DNN	Deep Neural Network
GPU	Graphics Processing Unit
ML	Machine Learning
SMOTE	Synthetic Minority Oversampling Technique
t-SNE	t-distributed Stochastic Neighbor Embedding

Table of Contents

1 Introduction

Machine learning algorithms have shown their strength and practicality in a myriad of day-to-day applications. The detection of fraud is an example of one application where machine learning and deep learning techniques have shined and demonstrated their usefulness, specifically by learning the intrinsic rules and properties that can be hidden in the data. We, therefore, chose the context of credit card fraud detection for our application since it not only presented a real-life problem where machine learning can be useful, but it also provided nice challenges to test out different algorithms and preprocessing applications seen in class, as it presents highly skewed and imbalanced data. The objective of this project is to use and compare the performance of deep learning and semi-supervised learning to detect fraudulent credit card transactions over non-fraudulent ones.

In this study, we first start by exploring how deep learning can help our application, as the use of deep neural layers can often allow the modeling of more complicated functions from our data which can help with complex data relations such as in fraud detection. We also compare the use of DNNs with and without the use of data augmentation techniques such as SMOTE and compared them to semi-supervised learning techniques. Finally, we explore how can the architecture of DNNs can be optimized using AutoML approaches, and how they can be useful in the creation of a deep learning model.

The rest of this paper is organized as follows: Section 2 presents some related publications on the topic, Section 3 displays information regarding the dataset used and our testing methodology, Section 4 presents the results obtained and their analysis, and finally, Section 5 presents our conclusions and main takeaways from this project.

2 Background

In other works also performing credit card fraud detection, we note the work of Dornadula and Geetha (2019) who used two random forest methods to learn the behavioral characteristics of normal and abnormal transactions: random-tree-based random forest and a CART-based random forest. On a limited dataset, the random forest strategy performed better, although the accuracy was reduced by unbalanced data. Additionally, Aleskerov, Freisleben, and Rao (1997) created CARDWATCH, a database mining system based on a neural network learning module. Past data from a single client was utilized to train a neural network, which is then used to analyze current spending patterns and spot anomalies. Awoyemi, Adetunmbi, and Oluwadare (2017) performed an analysis of more classical ML models, namely naïve Bayes, k-nearest neighbor, and logistic regression to detect credit card fraud. They were able to obtain good results using a hybrid under/over-sampling preprocessing technique, which is why we decided to use SMOTE preprocessing, in order to see how it would perform on this data, using DNNs instead of basic models.

Furthermore, using autoencoder neural networks, Legrand, Niepceron, Cournier, and Trannois (2018) obtained good prediction performance for anomaly detection in connected buildings, which we ultimately decided to incorporate in our experiments to see how it would perform with a similar task (anomaly detection), but in a completely different field.

Finally, Tiwari, Mehta, Sakhuja, Kumar, and Singh (2021) provided a comparative analysis of different fraud detection techniques across different datasets which helped in identifying the performance of different machine learning algorithms. This motivated us to conduct our experiments with ANN and a semi-supervised approach.

3 System Description

3.1 Dataset

The dataset used is comprised of more than 280,000 credit card transactions made by European cardholders over the course of two days in September 2013 and is credited to the Machine Learning Group at the Free University of Brussels (Dal Pozzolo, Caelen, Johnson, & Bontempi, 2015). All details of the cardholders have been anonymized and a total of 28 unnormalized principal components of these anonymized features are provided as different features. In addition, the time between transactions in seconds and the purchase amount are presented, presumably in euros. Each record is classified as either a regular (class "0"), or a fraudulent (class "1") transaction, with the number of transactions heavily favoring the former. As seen in Figure 1, out of a total of 284,807 transactions, only 492 are fraudulent credit card transactions, accounting for about 0.172 percent of all transactions. Moreover, Figure 2 displays the t-SNE representation of the fraudulent and benign transactions, which allows the representation of a multi-dimension dataset into a 2D plot. We performed this analysis during our preliminary data exploration in order to better understand the data we were dealing with and establish further steps to preprocess it.

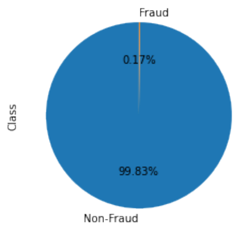


Fig. 1: Pie chart representing dataset imbalance

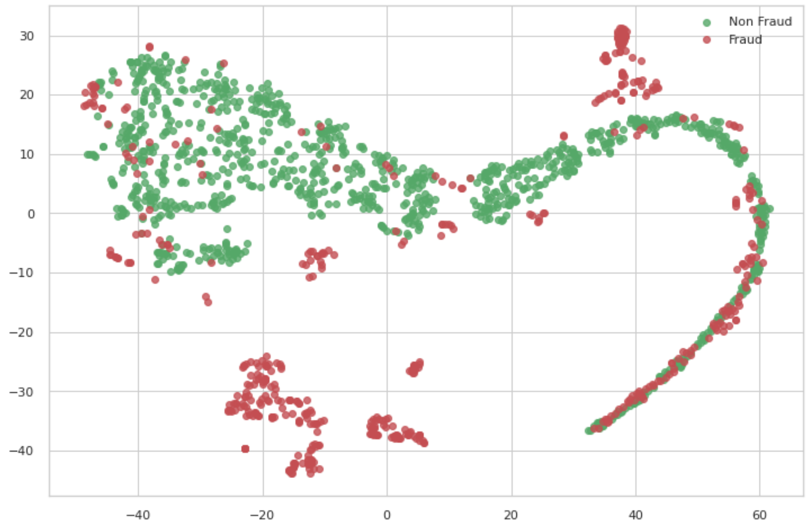


Fig. 2: t-SNE representation of the data

3.2 Data Preprocessing

Understanding the data's format and distribution is a crucial first step in any machine learning problem. It helps in figuring out where the most information is lying and to analyze outliers in a dataset. As we've mentioned prior, the dataset used displays heavily imbalanced classes and data that is not normalized, while on the positive side, not having any missing values. Due to these characteristics, we decided to perform two main types of preprocessing to deal with the skewness and render the data more Gaussian, namely skewness reduction and standardization, while to deal with the imbalance of the data, we performed SMOTE, a technique widely regarded as the standard when dealing with such data (Fernández, Garcia, Herrera, & Chawla, 2018).

3.2.1 Skewness Reduction

Skewness is defined as a measure of how far a probability distribution deviates from the symmetrical normal distribution (bell curve) in a given set of data. When talking about symmetrically distributed data, we are referring to the normal distribution, as it has zero skewness. A high level of skewness in the input variables is often regarded as undesirable, as it may violate model assumptions, reduce the interpretation of feature relevance, and ultimately lead to overfitting or even poor performance. For this reason, we decided to observe the level of skewness of our input variables and apply a power transform technique to features that displayed a skewness outside a given threshold [-1,1]. Consequently, this action was performed on 19 out of the 30 total features present in the dataset, which are represented in Table 1.

Feature	Skewness
V1	-3.280667
V2	-4.624866
V3	-2.240155
V5	-2.425901
V6	1.826581
V7	2.553907
V8	-8.521944
V10	1.187141
V12	-2.278401
V14	-2.278401
V16	-2.278401
V17	-3.844914
V20	-2.037155
V21	3.592991
V23	-5.87514
V27	-1.170209
V28	11.192091
Amount	16.977724
Class	-2.278401

Table 1: Features with a skewness outside [-1,1]

3.2.2 Standardization

Standardization is required when elements of an input data collection have significant disparities across their ranges, or simply when they are measured in multiple measurement units (e.g., pounds, meters, miles, etc.). Discrepancies in the ranges of initial features may cause trouble to several machine learning algorithms, including DL models, as it can make it difficult for the model to adjust the weights of its neurons, and ultimately render it unstable.

Data standardization is the process of re-scaling the attributes to have a mean of 0 and a variation of 1. Standardization's ultimate goal is to reduce all features to a single scale without distorting the differences in value ranges. In that regard, we performed standardization on the dataset with each feature centered and scaled independently of one another.

3.2.3 Data Imbalance

Data imbalance is a problem that plagues many ML real-world applications, as the relevant event is often the least occurring one (fraud, intrusion, anomaly, etc.). This skew in the classes causes many problems, as numerous ML algorithms struggle to learn from these rare events and even disregard them as unimportant in certain cases (Branco, Torgo, & Ribeiro, 2016).

Some of the simplest techniques to address this data imbalance are simply to duplicate the entries of the minority class or remove entries from the majority one, however, these techniques do not add any new information to the model. A more involved approach, introduced by Chawla, Bowyer, Hall, and

Kegelmeyer (2002) as SMOTE, is based on the fact that new data points can be synthesized to create new ones, which is performed to increase the number of samples of the minority class. SMOTE works by first selecting a class A randomly and finding the K nearest neighbors of that class. One of the neighbors will be selected, referred to as B. The two points, A and B will be connected to form a line segment in the feature space and which will be used to create the new synthetic instances (H. He & Ma, 2013). Using this technique, we were able to balance the classes equally which changed drastically the performance of our models, as we will demonstrate in the next section.

3.3 Performance Evaluation

Given the imbalance of the data and our preference bias on detecting the fraud cases rather than the benign ones, a common metric such as accuracy can't be used for this model, as the impact of our minority class will not be correctly represented in the metric value (Branco et al., 2016). Other metrics such as precision and recall are therefore more representative of the problem and should be used accordingly. Furthermore, even though we used both of the aforementioned metrics, for readability and ease of comparison between models, a metric described as the harmonic mean of precision and recall, namely the F1-Score, will also be used when comparing the models.

3.4 Methodology

Our approach to applying DL to this problem is divided into four main steps: (i) evaluate the performance of a baseline DL model on the problem, using only the skewness reduction and normalization techniques mentioned above; (ii) fit the same baseline model on the resampled dataset using SMOTE to balance the two classes; (iii) use AutoML techniques, as described below, to improve the model's architecture and hyperparameters in order to obtain better a better performance; and (iv) use autoencoders accompanied by a basic ML model. These four steps will allow us to compare the use of DL techniques with and without data rebalance and hyperparameters optimization, as well as test their use in encoding techniques.

3.4.1 Baseline Model

For our baseline model, we used binary cross-entropy as the loss function, as this is a binary classification problem, with the following architecture:

1. Input Layer (29 features input, 16 neurons)
2. Dense Layer (24 neurons, ReLU activation)
3. Dropout Layer (Frequency of 0.5)
4. Dense Layer (20 neurons, ReLU activation)
5. Dense Layer (24 neurons, ReLU activation)
6. Output Layer (Sigmoid activation)

We have experimented with different options for the architecture of our baseline model and ended up choosing the neural network architecture with three hidden layers, as per Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov (2014). We also introduced a dropout frequency of 0.5 to avoid overfitting and observed its significance while handling the unbalanced data. Finally, ReLU was chosen as an activation function for our hidden layers, as per other works we reviewed in Section 2 (Nwankpa, Ijomah, Gachagan, & Marshall, 2018).

3.4.2 AutoML

Hyperparameters and model parameters are the two different types of parameters that make up ML models. All of the parameters that the user can set freely before starting training are referred to as hyperparameters (eg. number of neurons in a neural network, learning rate, etc.), whereas model parameters are learned during model training (eg. weights in a DNN). A low number of neurons in a neural network can potentially lead to underfitting, while a high number can lead to overfitting. Those are both behaviors that we want to avoid for obvious reasons, therefore, we need to determine an appropriate number of neurons to use to get the best results possible. This makes hyperparameter tuning an essential component of every machine learning pipeline.

Moreover, every neural network has various hyperparameters such as the number of neurons per layer, number of layers, learning rate, momentum, initializers, activation functions, number of epochs, batch size of the input, optimizers, and much more. We need to find the best combination of values for all of these hyperparameters, which makes this process a very complex and time-expensive task. There are several ways to perform hyperparameter tuning, but most involve having a good grasp and knowledge of the field in which the algorithm is deployed. In our case, since we did not have extensive knowledge of the field of credit card fraud, and we wanted to further explore the power of automated techniques, we used an approach known as AutoML. These techniques encompass all methods which involve automatically exploring the search space of hyperparameters and architecture optimization (X. He, Zhao, & Chu, 2021). In our case, we made use of an open-source library called AutoKeras, which is a library built for DNNs and based on Keras.

3.4.3 Autoencoders

The other approach we followed for this problem is a semi-supervised classification approach using autoencoders. Semi-supervised learning is a combination of supervised and unsupervised learning processes. In this approach, the properties of unsupervised learning are used to learn the best possible representation of data, and the properties of supervised learning are used to learn the relationships in the representations which are then used to make predictions. One of the biggest challenges of this problem is that the target is highly imbalanced, as only 0.17% cases are fraud transactions. However, the advantage of the representation learning approach is that it is still able to handle the imbalanced nature of such problems. In the previously displayed Figure 2, we can observe that there are many non-fraud transactions very close to fraud transactions, which ultimately makes the classification very difficult. Autoencoders, which are a special type of neural network architecture in which the output is the same as the input, are well known to be able to deal with these kinds of problems. They are trained in an unsupervised manner in order to learn the extremely low-level representations of the input data. These low-level features are then deformed back to project the actual data.

We first created an autoencoder model in which we only show the model of non-fraud cases, and where it will try to learn the best representation of these cases. The same model will be used to generate the representations of fraud cases, as we expect them to be different from non-fraud ones. The beauty of this approach is that we do not need too many samples of data for learning good representations. In our case, we only used 2000 rows of non-fraud cases to train the autoencoder. Additionally, we do not need to run this model for a large number of epochs. The visualization below in Figure 3 shows the latent representation obtained from our autoencoder. We can now observe a clear discrepancy between fraud and non-fraud transactions, as well as being easily linearly separable. Consequently, we now do not need a complex model to classify this problem, in our case, we used a simple logistic regression model.

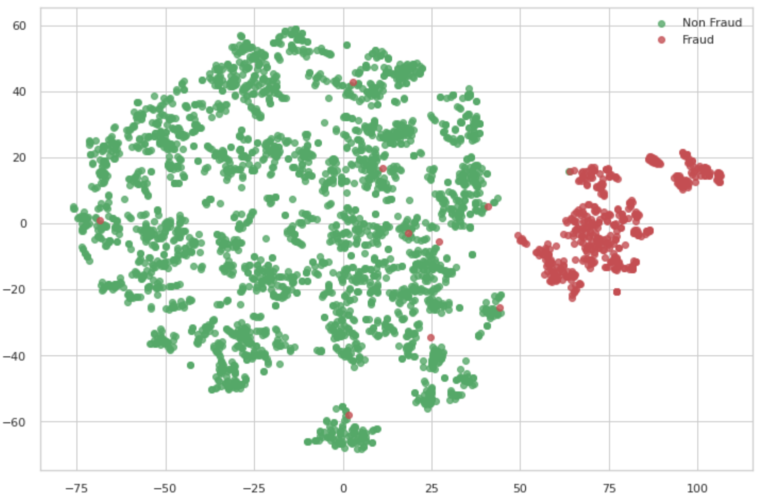


Fig. 3: Latent representation of the data using an autoencoder

3.5 Environment

Our models were built using Python version 3.7.12. Python was chosen for its simplicity and access to great libraries and frameworks for artificial intelligence and machine learning. For designing the DL models, we employed Keras, which is built on top of Theano and TensorFlow. Keras is an open-source Python library for developing and evaluating deep learning models. Finally, we used AutoKeras to optimize the DNN using AutoML.

All models were trained on Google Colab with the GPU runtime environment. For building the models, we used the sequential API-based Keras model, which allowed us to create a model layer-by-layer. Input data was split into training and testing datasets with a ratio of 8:2 and all models used binary cross-entropy as their loss function.

4 Numerical Results

Table 2 presents the results for both our baseline and AutoML optimized model, on the regular and SMOTE rebalanced dataset. This table will be referred to in the subsequent sections, as we will further discuss the results obtained by both models.

	Imbalanced Data			**SMOTE Resampled Data**		
	Precision	**Recall**	**F-1 Score**	**Precision**	**Recall**	**F-1 Score**
Baseline Architecture	0.856	0.769	0.810	0.996	0.999	0.998
AutoML Optimized Architecture	0.903	0.762	0.827	0.998	1.0	0.999

Table 2: Results from baseline and AutoML optimized models

4.1 Baseline Model

Looking at the results in Table 2, we can clearly see the baseline model underperforming when learning on the imbalanced data compared to the balanced one. However, even if the score is lower than its balanced counterpart, the model still achieved decent performance, with a F-1 score of 0.827. When balancing the data using SMOTE, we can observe a significant increase in performance where the model reaches a near-perfect score with a 0.998 F-1 score. This is also represented in the loss function and confusion matrix, as it can be seen in Figures 4 and 5. We expected this behavior when comparing balanced and imbalanced data since as we've mentioned in Section 3.2.3, ML models can have a difficult time learning from a class that doesn't appear as often. These results restate the efficiency of SMOTE as a balancing technique since as we saw, the model had much better performance on the minority class after resampling.

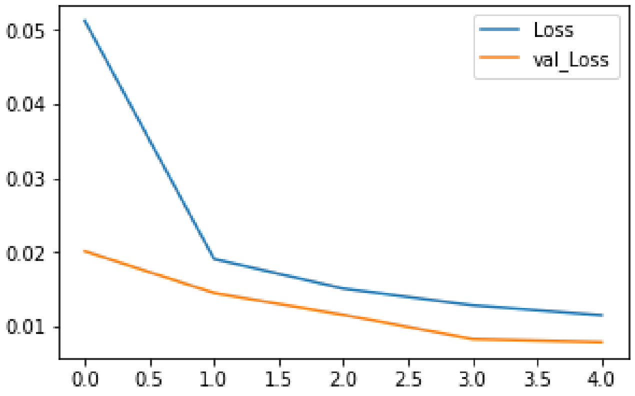
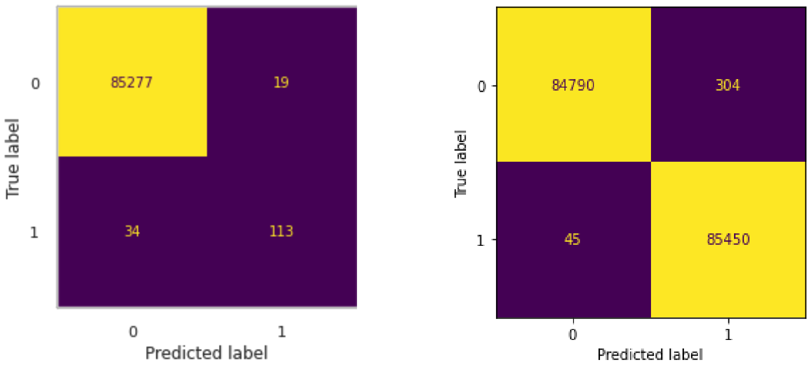


Fig. 4: Loss during training of the baseline model

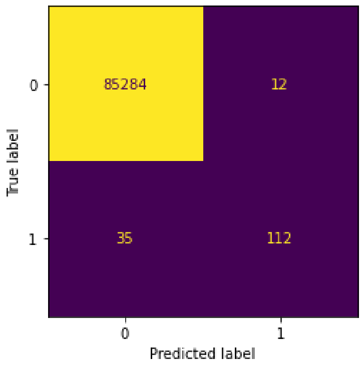


(a) Imbalanced data (b) SMOTE resampled data

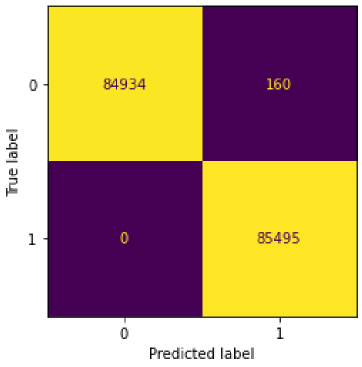
Fig. 5: Confusion matrices of baseline model

4.2 AutoML

The results of the use of the AutoML algorithm on both datasets, resampled or not, shown in Table 2 and its confusion matrices in Figure 6, show an increase in the overall score for both models. An interesting observation of the imbalanced data is that the optimized model mainly improved in precision rather than recall. This means that tweaking the hyperparameters allowed the model to perform better in the accuracy of the frauds it finds. The balanced dataset also shows a small increase in performance, albeit small given that the baseline performed nearly perfectly already, with an F-1 score of 0.999 compared to 0.998 for the baseline model. The fairly small performance increases could signify that our initial model architecture was fairly good and that the optimization required was minor. It is worth noting however that we only ran the AutoML optimization algorithm for a small number of trials (10) due to time constraints, as the optimization does take a very long time to process, so there might be further optimization possible which would lead to better results. This is one of the main drawbacks of using AutoML techniques, as they often do take a lot of time to optimize and are computationally intensive.



(a) Imbalanced data



(b) SMOTE resampled data

Fig. 6: Confusion matrices of the AutoML optimized model

4.3 Autoencoders

Table 3 displays the results obtained using the autoencoder technique we presented in Section 3.4.3. Given that these results were obtained on the raw, not resampled data, they are impressive, as they almost reached the F-1 score of the baseline model with resampled data (0.956 vs 0.998). Recalling that this approach uses a logistic regression model for the classification after the data was encoded using the autoencoder, shows the power and efficiency of autoencoder models in finding hidden representations in data with high dimensions and puts them as a valid alternative to resampling techniques.

Precision	**Recall**	**F-1 Score**
1.0	0.916	0.956

Table 3: Results from the autoencoder model

5 Conclusion

This paper presented the application of multiple DL techniques to the field of credit card fraud detection. With our results and analysis, we were able to test the efficiency of these techniques and reach interesting conclusions. Class imbalance is a common problem in multiple artificial intelligence fields, such as anomaly detection and fraud detection, and ends up causing DL models to generally underperform and have a difficult time classifying the minority class. During our experiments, we observed that without changing the model itself, we managed to substantially improve the model's performance by simply equalizing the class using SMOTE to synthetically increase the number of samples of our minority class.

Moreover, we note that optimizing the model's hyperparameters can have a drastic impact on the model's performance. In that regard, we explored AutoML techniques in order to optimize the architecture and hyperparameters of our model. While our baseline already performed well on both imbalanced and balanced data due to our manual tweaking, our results did show a performance improvement when using the AutoML optimized architecture, albeit small. We conclude that these techniques can be really useful for users with a lesser knowledge of a given field since they automate the process of tweaking the model's parameters to an extent. However, it is to be noted that, while useful, these techniques are computationally intensive, and still require a certain level of knowledge.

Thirdly, we also tested the use of neural networks in the form of autoencoders. We noted that not too many samples were required to fit the autoencoder and that once the data was encoded, it was fairly easy for a basic ML model to perform the classification. We also observed that the model trains quickly and that there was no need to train the encoder for a large number of epochs to achieve good performance. All in all, the autoencoder proved to be very effective at finding the hidden representations in the data and bringing them to a lower amount of dimensions which could then be easily classified.

In conclusion, once data rebalancing was performed on the data, we showed that it was fairly easy for a fairly basic DL model to be fitted and achieve good performance on this data. We note that performing these experiments on a more complex dataset, such as one presenting both imbalance and multiple classes, could be an interesting future application for this work.

References

Aleskerov, E., Freisleben, B., Rao, B. (1997). Cardwatch: A neural network based database mining system for credit card fraud detection. *Proceedings of the ieee/iafe 1997 computational intelligence for financial engineering (cifer)* (pp. 220–226).

Awoyemi, J.O., Adetunmbi, A.O., Oluwadare, S.A. (2017). Credit card fraud detection using machine learning techniques: A comparative analysis. *2017 international conference on computing networking and informatics (iccni)* (pp. 1–9).

Branco, P., Torgo, L., Ribeiro, R.P. (2016). A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys (CSUR)*, *49*(2), 1–50.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321–357.

Dal Pozzolo, A., Caelen, O., Johnson, R., Bontempi, G. (2015, 12). Calibrating probability with undersampling for unbalanced classification.. 10.1109/SSCI.2015.33

Dornadula, V.N., & Geetha, S. (2019). Credit card fraud detection using machine learning algorithms. *Procedia computer science*, *165*, 631–641.

Fernández, A., Garcia, S., Herrera, F., Chawla, N.V. (2018). Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research*, *61*, 863–905.

He, H., & Ma, Y. (2013). Imbalanced learning: foundations, algorithms, and applications. In (p. 47). Wiley-IEEE Press.

He, X., Zhao, K., Chu, X. (2021). Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, *212*, 106622.

Legrand, A., Niepceron, B., Cournier, A., Trannois, H. (2018). Study of autoencoder neural networks for anomaly detection in connected buildings. *2018 ieee global conference on internet of things (gciot)* (pp. 1–5).

Nwankpa, C., Ijomah, W., Gachagan, A., Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Tiwari, P., Mehta, S., Sakhuja, N., Kumar, J., Singh, A.K. (2021). Credit card fraud detection using machine learning: A study. *arXiv preprint arXiv:2108.10005*.