Credit Card Fraud Detection

1st Peter Jinson
Master of Data Analytics
Western University
London, Canada
sjinson@uwo.ca

2nd Zirong Liu
Master of Data Analytics
Western University
London, Canada
zliu659@uwo.ca

3rd Muhammad Saad
Master of Data Analytics
Western University
London, Canada
msaad63@uwo.ca

***Abstract*—In response to the global challenge posed by credit card fraud, which incurs significant financial losses, this study explores the efficacy of integrating advanced machine learning techniques into fraud detection systems. Traditional detection methods, which often fall short against sophisticated and evolving fraud strategies, are enhanced by a suite of machine learning models, including logistic regression, decision trees, random forests, and deep neural networks. These models are chosen for their ability to handle large, imbalanced datasets and their proficiency in recognizing subtle patterns indicative of fraudulent activity.**

In this research, our methodology is rigorously structured, beginning with a comprehensive data pre-processing stage that involves normalization, handling missing values, and encoding categorical features to prepare a simulated dataset that closely mirrors real transaction data. The core of the study involves training various machine learning models on this dataset, with a particular focus on feature engineering to improve model sensitivity to fraud indicators. Model performance is evaluated based on a range of metrics, including accuracy, precision, recall, and F1-score, to determine their effectiveness in a real-time fraud detection environment.

Results from the study highlight that ensemble methods, such as boosted trees and random forests, along with deep learning approaches, notably convolutional neural networks and recurrent neural networks, significantly outperform baseline models. These models not only excel in detecting fraudulent transactions with high accuracy but also adapt to new patterns of fraud, demonstrating robustness against the dynamic nature of credit card fraud.

In conclusion, the integration of machine learning into credit card fraud detection systems marks a significant advancement in combating financial fraud. Future work will focus on reducing false positives and incorporating real-time adaptive learning models to maintain and enhance detection efficacy. The lessons learned underscore the critical importance of continuous model training and the adaptive capabilities of machine learning systems to respond to evolving fraud tactics.

***Index Terms*—Credit Card Fraud Detection, Machine Learning, Random Forest, Deep Neural Networks, Artificial Intelligence, Banking, Finance**

I. Introduction

Credit card fraud remains a critical challenge within the global financial sector, incurring significant economic losses annually. Traditional fraud detection systems, predominantly rule-based and static, struggle to combat the sophistication and dynamic nature of modern fraudulent techniques. These systems are plagued by high false positive rates and a general inability to swiftly adapt to novel fraud patterns, leading to operational inefficiencies and considerable financial strain due to the rarity of fraudulent transactions in vast datasets.

In this research we aim to bridge the research gap by integrating advanced machine learning techniques into fraud detection processes to enhance autonomous detection capabilities and dynamic adaptability to evolving fraud strategies. By employing a range of machine learning models, including deep learning networks and ensemble methods, on a simulated dataset that mirrors real-world transactional activities, this research demonstrates significant improvements in detection accuracy and operational efficiency. The innovative application of these complex algorithms facilitates real-time transactional data processing, learning, and adaptation to emerging fraud patterns, marking substantial progress over conventional methods. Theoretically, this work enriches the academic discourse on the application of artificial intelligence in financial security, offering a novel framework for future investigations. Practically, it provides a scalable solution that can be integrated into existing systems to reduce financial losses and enhance customer trust and satisfaction. The structure of this report unfolds as follows: an introduction to the problem and research objectives; a review of literature underscoring current system limitations; a detailed methodology for data handling, model training, and evaluation; a presentation of research findings with a discussion on their implications; and a concluding chapter offering future research directions and practical applications of the study's outcomes.

II. Background and Related Work

A. *Maintaining the Integrity of the Specifications*

The field of credit card fraud detection has been extensively studied, driven by the increasing incidence of financial fraud and the corresponding need for more effective detection systems. Traditional fraud detection techniques primarily rely on rule-based systems, which are criticized for their inflexibility and high rates of false positives [1]. Recent advances in machine learning have prompted researchers to explore data-driven approaches to overcome these limitations.

Literature in the domain typically segments fraud detection techniques into two categories: statistical and machine learning-based approaches. Statistical approaches, such as logistic regression and anomaly detection, have been foundational in identifying inconsistencies in transaction data [2]. However, these methods often fail to capture complex patterns in data which are non-linear and interactive in nature, thus limiting their effectiveness against sophisticated fraud strategies.

Machine learning techniques, on the other hand, offer a more dynamic approach to fraud detection. A comprehensive analysis by Abdallah et al. [3] showcases various machine learning strategies that have been employed, including supervised techniques like decision trees and support vector machines, and unsupervised techniques like clustering. More advanced methodologies, such as ensemble methods and neural networks, have been shown to improve detection rates significantly by leveraging their ability to learn from vast amounts of data and identify subtle, non-intuitive patterns [4].

Despite these advancements, there remains a notable research gap in the real-time application of these models. Most studies focus on offline fraud detection, where models are trained and tested on historical data. The challenge of implementing these models in a real-time environment, where they must make instant decisions on new and unseen data while adapting to evolving fraud tactics, is not adequately addressed [5]. Moreover, few studies explore the integration of these techniques into existing financial infrastructures, which is critical for practical deployment.

Additionally, the field lacks comprehensive comparative studies that evaluate the performance of various machine learning techniques against one another in identical conditions. Such studies are essential to understand the relative strengths and weaknesses of different approaches, particularly in terms of scalability, adaptability, and computational efficiency [6].

In summary, while the literature provides a robust foundation of methods for detecting credit card fraud, significant gaps remain in applying these techniques in real-time scenarios and integrating them seamlessly with existing financial systems. Future work must focus on these areas to develop more effective, adaptable, and efficient fraud detection systems.

III. Dataset

We downloaded the dataset from Kaggle, provided by the user 'Machine Learning Group - ULB - Andrea' [7]. The dataset consists of credit card transactions made by European cardholders in September 2013, spanning two days and containing 284,807 transactions, of which 492 were identified as fraudulent. This results in a highly unbalanced dataset, with fraudulent transactions accounting for only 0.172% of the total. All input variables, except 'Time' and 'Amount', underwent a PCA transformation, resulting in features V1 through V28. The 'Time' feature captures the time elapsed between each transaction and the first in the dataset, while 'Amount' indicates the amount of the transaction, useful for cost-sensitive learning. The response variable 'Class' distinguishes fraudulent transactions from legitimate ones. Due to the class imbalance of the dataset, it is recommended to measure model performance using the area under the precision-recall curve, as the accuracy of the confusion matrix can be misleading.

IV. Methods

A. Logistic Regression

1) Research Objective ID 01:
Classify fraudulent transactions using a Logistic regression model.

2) Research Methodology ID 01:
Logistic regression is used to model the probability of a fraudulent transaction using the several input variables available in the dataset. Logistic regression commonly models binary responses such as classifying a transaction as fraudulent or not. Classifying the data used the LogisticRegression() package from sklearn. Prior to model training, preprocessing steps are taken to remove erroneous data. The dataset is originally standardized and thus removing the need for applying standardization once again. Given the inherent highly imbalanced dataset, SMOTE (Synthetic Minority Over-Sampling Technique) is is used to generate artificial instances of the underrepresented response class. The model is then trained on the balanced data set randomly split into a 80% training and 20% test set. Using this training data, the logistic regression model is then fitted to the training data using Maximum Likelihood Estimation. To promote model robustness, 10-fold cross validation is used during the training phase.

B. Random Forest

1) Research Objective ID 02:
Applying Random Forest Classifier to detect fraudulent transactions

2) Research Methodology ID 02:
Next, a Random Forest Classifier is implemented to recognize fraudulent transactions in the dataset. Random Forest builds a number of trees to their maximum depth to result in a collection of diverse trees. The final prediction is made by taking the majority vote among all the built trees. Similar to logistic regression, data is randomly split into 80% training and 20% testing sets to construct the model. To account for the data imbalance, BalancedRandomForestClassifer() from imblearn.ensemble is used to draw bootstrap samples from the minority class and sample with same number from the majority class.

C. Gradient Boost

1) Research Objective ID 03:
Build boosted trees to classify fraudulent transactions with higher accuracy.

2) Research Methodology ID 03:
Gradient Boosting is also used to sequentially built trees using corrections from previously built trees. Correlation between the available classes is assessed to identify potential stumps that required additional weight to be applied. As opposed to Random Forest Classification, Gradient Boosting makes the classification is made by summing the predictions of all trees with each tree's contribution accounted for by their respective weights.

D. Support Vector Machine

1) Research Objective ID 04:
To detect credit card fraud using Support Vector Machine (SVM).

2) Research Methodology ID 04:
This research employs Support Vector Machine (SVM) to develop a robust fraud detection system using a highly unbalanced dataset characteristic of real-world financial transaction data. The data encompasses transactions where, except for the transaction time and amount, all other features are anonymized due to privacy reasons and have undergone Principal Component Analysis (PCA) transformation, a common dimensionality reduction technique. It is important to note that for effective PCA transformation, features must be scaled, which we assume has been adequately handled in the dataset preparation phase.

We initiated the study by loading transaction data into a Jupyter notebook, converting it into a Pandas Data Frame to facilitate data handling and visualization. Our initial exploratory analysis provided a basic understanding of the data's structure, the summary statistics are listed below:

1. The transaction amounts are relatively small, with an average of approximately USD 88.

2. The dataset contains no null values, obviating the need for imputation techniques.

3. A vast majority of the transactions are non-fraudulent, occurring 99.83% of the time, while fraudulent transactions represent a mere 0.17%.

Given the vast discrepancy between non-fraudulent and fraudulent transactions, we opted for under-sampling the majority class to balance the dataset. This approach was chosen over oversampling to avoid the introduction of excessive synthetic data, which could potentially skew the model's ability to generalize from genuine transaction patterns.

The SVM model was first trained using a linear kernel to establish a baseline. Adjustments to the class weights were made to improve the detection sensitivity towards fraudulent transactions. We experimented with various kernels—linear, polynomial, radial basis function (RBF), and sigmoid—to compare their efficacy.

E. Feed-forward Neural Network

1) Research Objective ID 05:
In addition to exploring various fraud detection algorithms, we implemented a simple feed-forward neural network (FFNN) to demonstrate the feasibility and effectiveness of neural networks in fraud detection by directly comparing the model's performance to industry benchmarks such as tree-based ensembles.

2) Research Methodology ID 05:
To address the problem of credit card fraud detection, we developed a Feed-forward Neural Network (FFNN) pipeline using the PyTorch library. The first step is to efficiently manage the dataset and transform it into a format compatible with PyTorch. The dataset consists of features labeled 'V1' to 'V28', 'Time' and 'Amount' as inputs, and 'Class' as output label. It is split into training and test sets in an 80:20 ratio using the train-test split function to ensure similar distributions of class labels between the sets.

To address the issue of class imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) is applied to generate synthetic samples for the minority class, balancing the training set. In addition, the numerical features 'time' and 'amount' are scaled using a standard scaler, ensuring a balanced distribution and facilitating model training.

The training loop is configured with a batch size of 64 and iterates over epochs. During each epoch, batches are fetched, forward passes are performed, losses are calculated, and model weights are updated using an optimizer. The shuffle parameter is set to True for the training data to ensure that the order of data points differs between epochs, thus preventing overfitting.

The next step is to design the neural network model. This involves implementing a simple FFNN using the PyTorch library. The model is built as a torch.module with an initialization function that sets up its layers and a forward function that defines its forward pass operations. The model consists of several fully connected layers using torch.nn.Linear. The first layer, fc1, has as many neurons as there are features in the input vector. This is followed by a hidden layer with a specified number of neurons (hidden size), using a Rectified Linear Unit (ReLU) activation function for nonlinear transformations. Finally, the output layer contains a single neuron with a sigmoid activation function to produce a binary classification output that labels transactions as fraudulent or legitimate.

The choice of a ReLU activation for the intermediate layers is based on its empirical performance in terms of optimization speed and effectiveness. The sigmoid activation for the output neuron is chosen for its ability to produce probabilities between 0 and 1, suitable for binary classification tasks.

To facilitate training and evaluation, the model is loaded into the chosen device (either CPU or CUDA, depending on availability). It then runs through training and evaluation cycles, using a data loader to batch, mix, and prepare the data for each iteration. This setup ensures efficient training, evaluation, and effective model generalization across epochs.

To optimize the model for detecting fraudulent transactions, we implement a training loop that iterates over multiple epochs, minimizing the discrepancy between the predictions and the ground truth labels. The model is designed to output a value between 0 and 1, representing the probability that a transaction is fraudulent or genuine. The Binary Cross Entropy (BCE) loss function measures this discrepancy and provides a criterion for training the model.

The training loop consists of performing forward passes for each batch of training data, computing the loss using BCE, and then performing a backward pass to update the model weights using the SGD optimizer. Over 150 epochs, training and validation losses are recorded, providing insight into the evolution of the model. The training loss shows a steady decrease, indicating successful optimization. The validation loss initially follows a similar trend, but then plateaus and oscillates, indicating potential over-fitting.

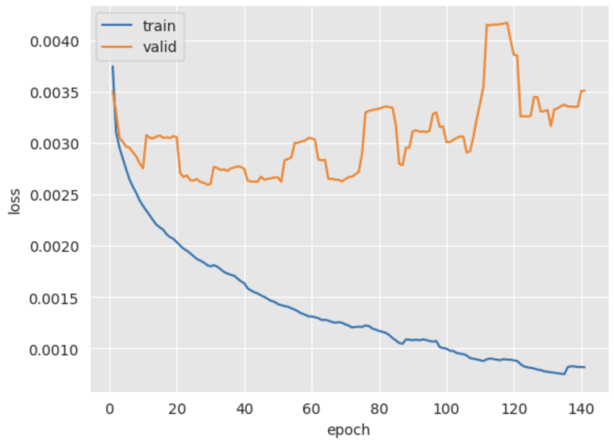

Fig. 1. Loss over Epochs for Training and Validation Sets

Figure 1 plots the loss over epochs for the training and validation data sets. The blue line represents the training loss, while the orange line represents the validation loss. The steady decrease of the training loss indicates a successful optimization. However, the validation loss shows oscillations and plateaus, indicating potential overfitting. This illustrates the balance between minimizing training loss and achieving generalization to unseen data.

To evaluate the performance of the model on unseen data, we evaluate its predictions on the test set by comparing the predicted values to the actual labels. Metrics such as AUC ROC, average precision, and map precision at various thresholds are calculated. The final model achieves an AUC ROC of 0.977, an average precision of 0.849, and a card precision@100 of 0.828, demonstrating strong performance in discriminating between fraudulent and genuine transactions, precision in predicting positive cases, and effective ranking of predictions.

To address the overfitting problem and improve convergence, an early stopping strategy is implemented. This approach monitors the validation loss and stops the training process if the loss increases consistently for a defined number of iterations, known as the patience parameter. This patience parameter ensures that small variations in the validation loss do not prematurely stop the training process.

Several optimization techniques are used to speed up convergence and achieve a reasonable extremum. First, the model is trained using SGD with a learning rate of 0.0005, which shows gradual improvements. Then, an adaptive learning rate optimizer, Adam, is used, which normalizes the learning rate based on the gradient norm, allowing for faster convergence and a more stable optimum.

Choosing Adam as the optimizer, coupled with tuning the batch size and initial learning rate, provides a balanced approach to convergence speed and optimal quality. In addition, techniques such as batch normalization and reducing the learning rate at validation loss plateaus can further improve performance, making neural network optimization a dynamic and evolving field.

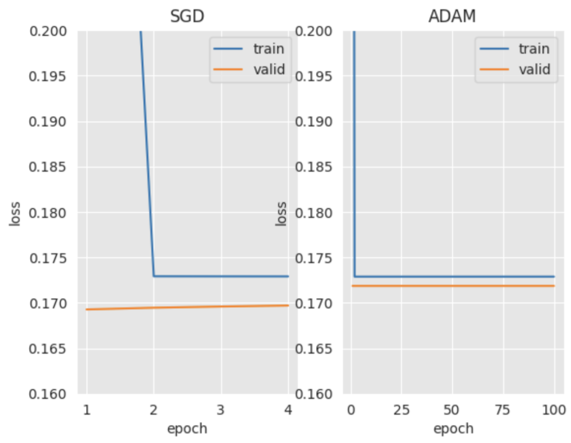

Fig. 2. Training and validation loss: Optimizer SGD vs Optimizer Adam

Figure 2 highlights the difference between training with SGD and Adam optimizers, showing how the latter leads to faster convergence and a better optimum.

In the left plot, we observe that SGD starts with a significant initial drop in loss, especially in the first epoch. However, the subsequent progress is gradual, with both training and validation losses stabilizing at similar levels. In contrast, the right plot shows that Adam's adaptive learning rate facilitates rapid convergence. In a fraction of the epochs compared to SGD, Adam achieves a lower and more stable loss for both training and validation sets, indicating its ability to quickly navigate to a better optimum.

The graph highlights Adam's advantage in achieving faster convergence and lower loss compared to SGD. This is particularly beneficial for deep learning models, where optimizing for speed and stability is critical. Adam's adaptive learning rate normalizes updates based on the gradient norm, allowing it to efficiently adapt to varying steepness levels during training.

This comparative analysis demonstrates the advantages of Adam, which are maintained throughout the rest of the chapter. Its adaptive nature reduces the need for extensive hyperparameter tuning and provides balanced performance across different models. Future tuning will focus primarily on batch size and initial learning rate to further refine the model's performance.

A classic way to improve generalization and achieve better validation performance is to regularize the model. One regularization technique specifically designed for neural networks is dropout. Dropout involves randomly dropping neurons from the network at each training step, creating a random subnetwork.

Figure 3 shows the effect of dropout on training and validation losses. The plot shows that with dropout (green and red lines), both losses stabilize quickly, indicating a balanced training process. Without dropout (blue and orange lines),

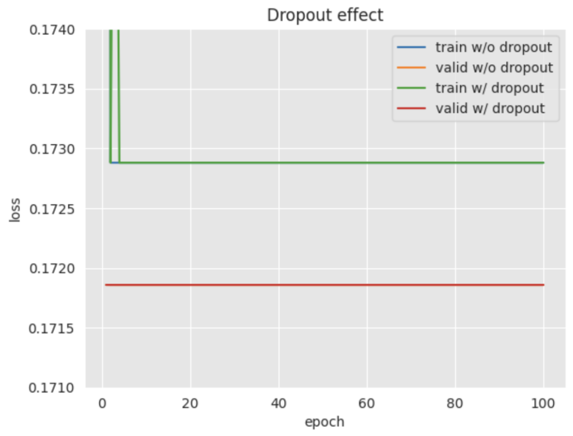

Fig. 3. Loss curves for the FFNN model with and without dropout regularization

training and validation losses remain relatively stable, but do not improve significantly, indicating potential over-fitting.

The figure illustrates how dropout can effectively regularize the model, ensuring that it generalizes better to unseen data. In practice, tuning the dropout parameter serves as a key hyper-parameter search, allowing for balanced model training and improved performance.

Dropout and other regularization techniques significantly improve generalization and model training. In particular, dropout reduces over-fitting and mimics ensemble strategies by creating diverse sub-models during training. This regularization approach, along with parameter tuning, ensures that our model remains robust and generalizes well across datasets.

Because our dataset was scaled and categorical variables were removed by the dataset provider due to privacy concerns, the implementation of scaling and embedding was not necessary.

V. Results

A. *Logistic Regression*

Like the training set, the test set is also subject to preprocessing methods including PCA and standardization. Figure 4 shows the results of testing the logistic regression model against the test data.

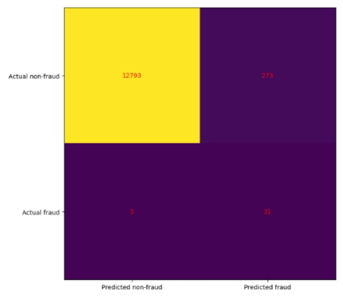

Fig. 4. The extreme imbalance is evident in this confusion matrix

The imbalance in data is apparent with the vast majority of the cases being non fraudulent. Likewise, using ROC (Receiver Operating Characteristics) yielded an AUROC (Area Under Receiver Operating Characteristics) of over 99% before an after applying SMOTE on the training data. This measure is quickly identified to be optimistic and quite ineffective. Given the interest of this study to identify the rare cases of fraud (positive class), Precision Recall (PR) Curve are of greater interest as it focuses on the rare yet important positive class. These metrics are key in applications where correctly identifying positive instances is crucial. Using the PR Curve, Figure 121234 shows the accuracy of predicting fraudulent cases correctly is calculated to 0.670 or approximately 67%. Logistic Regression is a simple classification technique and a relatively low prediction accuracy is therefore expected.

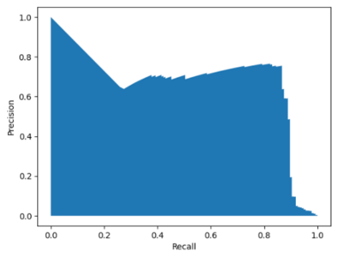

Fig. 5. Logistic Regression Classifier return an AUC of 0.67

B. *Random Forest*

Random Forest is produced using BalancedRandomForestClassifier() from imblearn.ensemble. A balanced forest draws a sample from the minority with the same replacement number of samples from the majority. Ensuring the model is trained on both classes equally can mitigate effects of class imbalance. The overall performance improvement of the classifier is evident by the PR curve in Figure 6. Random forest classifier improves on Logistic Regression with AUC of 0.841 or approximately 84.1%.

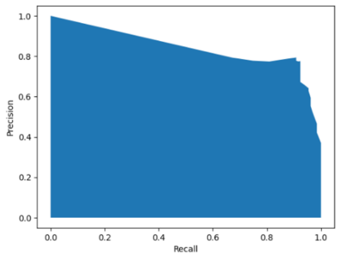

Fig. 6. BalancedRandomForestClassifier() return an AUC of 0.84

C. *Gradient Boosting*

Gradient Boosting Classifier improves on accurately predicting fraudulent transactions compared to Random Forest Classifier. While both algorithms performed admirably, gradient boosting exhibited superior performance in terms of predictive accuracy and robustness. Gradient Boosting, through its sequential training approach, effectively minimized errors and continuously improved model performance by iteratively correcting the mistakes of the previous iterations. This adaptability enabled the boosted classifier to capture patterns in the data and achieve higher precision and recall rates compared to Random Forest. With the superior features, Gradient boosted classifier yields an AUC of 0.99 or approximately 99%. This higher score is notable with considerations to a possibility of overfitting.

D. Support Vector Machine

The polynomial kernel demonstrated the highest accuracy at 99.79% as shown by the confusion matrix (Fig x), significantly surpassing the linear kernel's 95.94%. In contrast, the sigmoid kernel underperformed, achieving only a 66.75% accuracy, likely due to the non-linear nature of the data. Model performance was rigorously evaluated through confusion matrices and accuracy metrics, both during training and testing phases. The outcomes are depicted in visual data representations, which include confusion matrices before and after class-weight adjustments. This comprehensive analysis not only delineates the operational characteristics of each kernel but also underscores the importance of selecting appropriate machine learning strategies based on data distribution.

Potential threats to validity, such as overfitting due to high data dimensionality relative to the prevalence of fraud cases, were critically assessed. This research contributes significantly to both academic knowledge and practical applications in finance and cybersecurity, offering insights into optimizing fraud detection systems in increasingly digital financial landscapes.

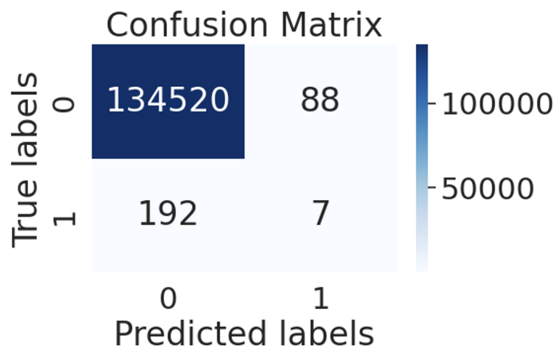

Fig. 7. The confusion matrix after changing the SVM kernel to a Polynomial Kernel

E. Feed Forward Neural Network

For the Feed-forward Neural Network (FFNN), initial training results indicate that the FFNN achieved high performance metrics, including an AUC ROC of 0.977, an average precision score of 0.849, and a precision@100 of 0.828. These metrics demonstrate the model's effectiveness in distinguishing between fraudulent and genuine transactions and providing accurate classifications.

Training and validation loss plots, as well as a comparative analysis of the SGD and Adam optimizers, show that Adam provides faster convergence and a better optimum, achieving comparable performance in significantly fewer epochs. This is consistent with Adam's ability to adapt the learning rate to gradient changes, allowing for efficient training.

Dropout regularization was also applied, effectively preventing overfitting and improving the model's generalization. The dropout effect graph illustrates this, showing balanced training and validation loss curves with regularization, compared to steady but unimproved losses without it.

However, the FFNN took a relatively long time to run, making it a costly model to run. Further parameter tuning and techniques are needed, as hyper-optimization of the architecture, activations, loss function, optimizers, and preprocessing comes at a cost. However, there are many ways to automate hyper-optimization and architecture design, such as using AutoML, including Neural Architecture Search. In addition, using data sets without PCA transformation, scaling, or ensuring more balanced data sets can help to more accurately reflect the performance of the FFNN.

VI. Conclusion & Future Work

A. Conclusion

This study examined and evaluated several credit card fraud detection algorithms, providing a comprehensive comparison of different models. The results of the study highlight both the strengths and limitations of each approach.

Logistic regression served as the base model, providing simple and interpretable results. However, its linear nature limited its ability to handle the complex, non-linear relationships inherent in fraud detection tasks, making it less effective for real-world scenarios.

The support vector machine model, particularly with a polynomial kernel, demonstrated high accuracy, reaching 99.79%. This suggests its robustness, especially when dealing with complex data structures. However, its effectiveness was limited by the dimensionality of the data and the model's lack of adaptability.

Random Forest, an ensemble method, showed balanced performance and significantly improved class imbalance through its bootstrap sampling technique. Its AUC ROC of 0.841 marked a significant improvement over logistic regression, demonstrating its potential for practical fraud detection applications.

Gradient Boost provided reliable results by using iterative corrections from previous learning steps. Its effectiveness varied depending on hyperparameter tuning and dataset characteristics, indicating the need for careful configuration to achieve optimal results.

Finally, the feedforward neural network achieved strong performance metrics, including an AUC ROC of 0.977, an average precision score of 0.849, and a precision@100 of 0.828. Its adaptive architecture, optimizer selection, and regularization techniques contributed to its success. However, its training process proved to be time consuming, indicating potential for further optimization.

B. Future Work

Future research can build on these results in several ways. First, model refinement through further tuning of hyperparameters, especially for ensemble methods and neural networks, can improve convergence speed and model performance. Integrating tools such as AutoML and Neural Architecture Search can streamline hyperoptimization, improving both accuracy and efficiency.

Next, incorporating data sets without PCA transformation, scaling, or ensuring balanced data sets can better reflect real-world scenarios. This approach can lead to more accurate models and robust detection capabilities.

Finally, exploring additional algorithms such as recurrent neural networks and convolutional neural networks can provide further insight into advanced fraud detection methods. Regularization techniques and adaptive optimizers can improve model generalization and provide a balance between performance and complexity.

C. Lesson

Several key findings emerged from this study. Balanced data sets, effective preprocessing, and comprehensive evaluation metrics are critical for accurate fraud detection.

Handling class imbalance through techniques such as SMOTE and balanced sampling is essential to ensure accurate models that effectively detect fraud cases.

Regularization techniques and adaptive optimizers, such as Adam, enhance model generalization, reduce overfitting, and improve performance on unseen data.

Finally, continuous improvement of fraud detection models is necessary, emphasizing strategies to counter evolving fraud tactics and protect financial systems.

In summary, this study contributes to the field of fraud detection by providing a comparative analysis of different algorithms, identifying areas for refinement, and highlighting strategies for future research.

References

[1] Bolton, R. J., and Hand, D. J., Statistical fraud detection: A review. Statistical Science, 2002, pp.235–249.

[2] - Phua, C., Lee, V., Smith, K., and Gayler, A comprehensive survey of data mining-based fraud detection research. Artificial Intelligence Review, 2010, pp. 231–263.

[3] Abdallah, A., Maarof, M. A., and Zainal, A., Fraud detection system: A survey. Journal of Network and Computer Applications, 2016, pp.90–113.

[4] Dal Pozzolo, A., Caelen, O., Le Borgne, Y. A., Waterschoot, S., and Bontempi, G.,Learned lessons in credit card fraud detection from a practitioner perspective. Expert Systems with Applications, 2015, pp. 4915–4928.

[5] Bhattacharyya, S., Jha, S., Tharakunnel, K., and Westland, J. C., Data mining for credit card fraud: A comparative study and Decision Support Systems, 2011, 50(3), pp.602–613.

[6] Carneiro, G., Figueiredo, M., and Costa, M., A data-driven approach to fight banking phishing using machine learning: Infosecurity Journal, 2017, pp.1–12.

[7] ML Group - ULB, ”Credit Card Fraud Detection,” Kaggle, Available online: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data, Accessed April 30, 2024. Alenzi, Hala, and Nojood Aljehane. 12th ed., vol. 11, International Journal of Advanced Computer Science and Applications, Tabuk City, 2020, Fraud Detection in Credit Cards Using Logistic Regression. Brownlee, Jason. “Repeated K-Fold Cross-Validation for Model Evaluation in Python.” MachineLearningMastery.Com, 26 Aug. 2020, machinelearningmastery.com/repeated-k-fold-cross-validation-with-python/. Chugh, Vidhi. “Precision-Recall Curve in Python Tutorial.” DataCamp, DataCamp, 19 Jan. 2023, www.datacamp.com/tutorial/precision-recall-curve-tutorial.