

Neural Network Models for Credit Card Fraud Detection

This project focuses on developing and evaluating **neural network models** for **credit card fraud detection**. A primary challenge is the **severe class imbalance** in the dataset, where fraudulent transactions are significantly outnumbered by legitimate ones. The methodology involves preprocessing the transaction data, specifically handling the class imbalance using **SMOTE and Tomek Links** techniques. Sequential neural network models are built and trained, experimenting with different activation functions and optimizers to learn complex patterns. Model performance is rigorously evaluated using metrics tailored for imbalanced data, including accuracy, precision, recall, F1 score, and the **Area Under the Precision-Recall Curve (AUPRC)**. The objective is to effectively train models for accurate fraud detection despite the imbalance and compare the impact of various model configurations.

Keywords: Credit Card Fraud, Fraud Detection, Neural Networks, Class Imbalance, Pipeline, SMOTE, Tomek Links, AUPRC, Evaluation Metrics.

 by T. Vinita

Introduction

Background and Motivation

Credit card fraud is a significant problem that costs billions of dollars annually[2][6]. Projections indicate the value of fraudulent card transactions worldwide is set to increase [6]. For businesses and financial institutions, the impact is severe [6][7]. This includes **financial loss** from chargebacks, transaction fees, and lost inventory [6][7]. Fraud can also lead to **damaged reputation**, as customers may avoid businesses where they were defrauded [6][7]. Furthermore, frequent fraud or chargebacks can result in **increased fees or restrictions from payment processors [7]**, and resolving cases requires time and resources, causing **operational disruptions [7]**. Fraudsters are motivated by financial gain, viewing credit card fraud as an easy way to steal money [6]. The financial industry faces this escalating threat globally, with North America identified as a major epicenter [4].

Problem Statement

A primary challenge in credit card fraud detection is the severe class imbalance in transaction datasets [2][3][8][4]. Fraudulent transactions represent a very small fraction of the total transactions, making them difficult for detection models to accurately identify [2][3][8]. For example, one dataset shows that out of 284,807 transactions, only 492 were fraudulent, meaning the positive class (frauds) accounts for only 0.172% of all transactions [3][2]. This imbalance can bias models towards the majority class (legitimate transactions), reducing their effectiveness in detecting fraud [2][8][4].

Furthermore, the evolving nature of fraud means fraudsters constantly refine their tactics to blend in with legitimate customer behaviour, demanding that detection systems be flexible and quick in adapting to new patterns [4]. Traditional fraud detection methods, such as Rule-Based Systems, Statistical Methods, and earlier Machine Learning (ML) Methods, have significant limitations and shortcomings, including missing new fraud patterns, generating high rates of false positives, being rigid, and requiring continuous manual updates [2][8].

The objectives of the process are primarily focused on training and evaluating neural network models for credit card fraud detection:

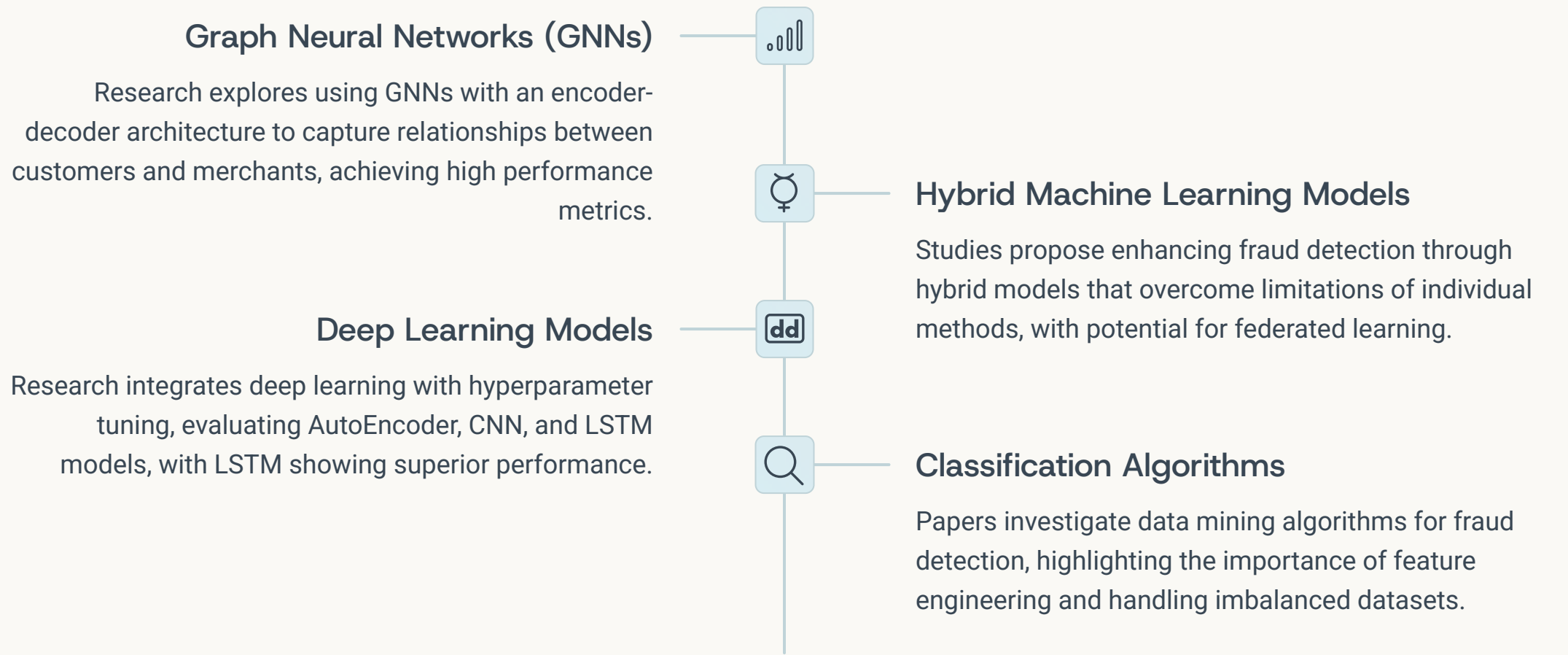
- To train neural network models to learn complex, non-linear patterns in credit card transaction data. This involves using activation functions to introduce non-linearity.
- To effectively train the model to make accurate predictions, particularly in detecting fraudulent transactions.
- To address the significant class imbalance issue in the dataset, where fraud cases are a very small percentage. Techniques like SMOTE and Tomek Links are used to create a more balanced dataset to avoid biasing the classifier towards the majority class.
- To evaluate the performance of the trained models using metrics such as accuracy, precision, recall, and F1-score, and visualization techniques like Precision-Recall curves. Area Under the Precision-Recall Curve (AUPRC) is used as a key performance metric.
- To compare the impact of using different activation functions (ReLU, Tanh) and optimizers (SGD, Adam, RMSprop) on model performance.
- To visualize results and model performance through plots of accuracy and loss over epochs, Precision-Recall curves, and metric scores to facilitate comparison between different model configurations.

This project is focused on using the "creditcardfraud" dataset where the target variable 'Class' distinguishes between genuine and fraudulent transactions. Handling the imbalance in such datasets is noted as crucial for credit card fraud detection.

Literature Review

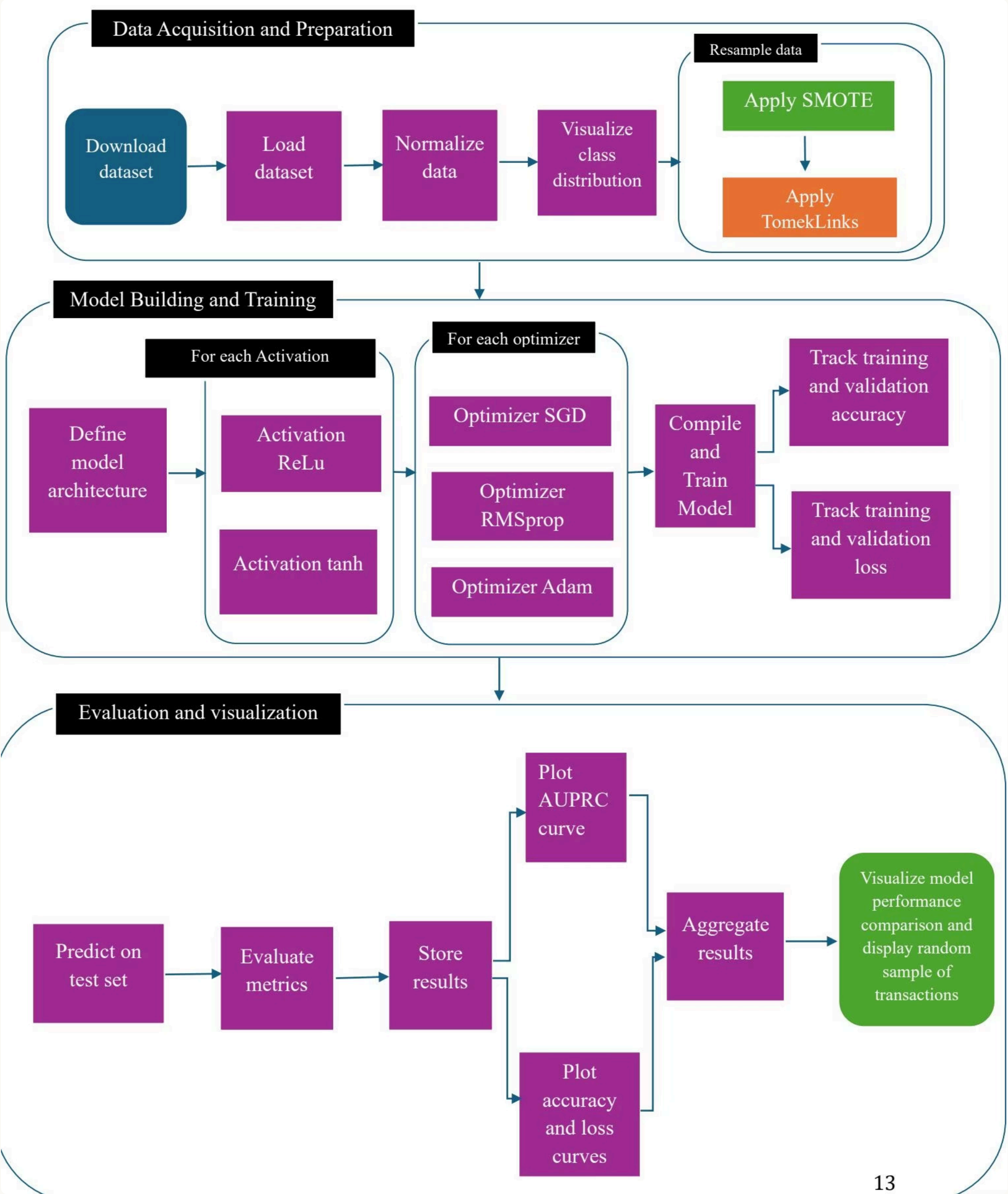
Model/Technique	Key Results/Metrics	Strengths	Weaknesses
Encoder-decoder GNN	Precision: 0.82, Recall: 0.92, F1: 0.86, AUC-ROC: 0.92	Captures complex relationships, effective	None explicitly mentioned
Hybrid ML Models	Not provided	Balances interpretability & performance	None explicitly mentioned
Improved Deep Learning Models	LSTM: Accuracy 98.3%	Surpasses previous work, trade-off between detection & precision	Need to enhance detection rate
Classification Algorithms	All data mining algorithms outperformed current system	Simple, interpretable, effective	Feature building and model tuning
Logistic Regression	Not provided	Effective for imbalanced datasets	Data imbalance
Ensemble CNN-LSTM	Early fusion performed better	Combines CNN & LSTM strengths	Can flag valid transactions or omit fraudulent ones
NN, k-NN, SVM, Decision Tree	NN, Decision Tree, k-NN showed high accuracy	High accuracy for imbalanced datasets	SVM: low accuracy for imbalanced datasets
Logistic Regression, Isolation Forest, K-Means, CNN	Decision Tree: Accuracy 94.87%, ROC: 0.87	DL models outperform traditional models for imbalanced data	Class imbalance, need for uniform data


The literature review reveals several key approaches to credit card fraud detection:



Compared to the approaches in the literature, our model uses a Dense Neural Network architecture with specific focus on activation functions (ReLU, Tanh) and optimizers (SGD, Adam, RMSprop). While the literature explores a wider range of models including GNNs, CNNs, and LSTMs, our approach emphasizes practical implementation with SMOTE and Tomek Links for handling class imbalance, which aligns with techniques mentioned in several studies. Our evaluation metrics (accuracy, precision, recall, F1-score, AUPRC) are consistent with those used in the literature, particularly for imbalanced datasets.


System Design and Methodology






Dataset Description

The project utilises the "creditcardfraud" dataset from Kaggle (user "mlg-ulb"), containing European cardholder transactions from September 2013 over two days. It includes approximately 284,807 total transactions, with only 492 fraudulent ones (0.172%). Features include PCA transformation results (V1 to V28), Time, and Amount, with the target variable 'Class' (0 for genuine, 1 for fraud).




Exploratory Data Analysis

The primary EDA step involves visualizing the class distribution using Seaborn's countplot to create a bar chart showing the count of samples in each class, highlighting the imbalanced nature of the dataset.



Preprocessing Steps

Data is loaded using `kagglehub.dataset_download()` and normalized with `MinMaxScaler` to scale features between 0 and 1, helping with faster convergence during neural network training. To address class imbalance, a pipeline combining SMOTE (over-sampling minority class) and Tomek Links (under-sampling majority class) is applied.



Model Architecture

Sequential neural network models are built with TensorFlow and Keras, consisting of an input layer, two dense hidden layers (128 and 64 neurons), and a single-neuron output layer with sigmoid activation. Different activation functions (ReLU, Tanh) and optimizers (SGD, Adam, RMSprop) are tested.

The activation functions are crucial as they introduce non-linearity into neural networks, allowing them to learn complex patterns:


ReLU (Rectified Linear Unit)

This function outputs the input directly if it's positive, and zero otherwise ($\text{ReLU}(x)=\max(0,x)$). It often tends to converge faster and reduces the risk of vanishing gradients.

Tanh (Hyperbolic Tangent)


This function outputs values between -1 and 1 ($\tanh(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$). It is zero-centered, which can sometimes be beneficial, especially if data is normalized around zero.

Optimizers are algorithms that guide the network's learning process by updating weights and biases to minimise the loss:




SGD (Stochastic Gradient Descent)

A basic algorithm that updates weights based on the gradient calculated on a single data point or a small batch. It can be slow to converge but can potentially escape local minima.



Adam (Adaptive Moment Estimation)

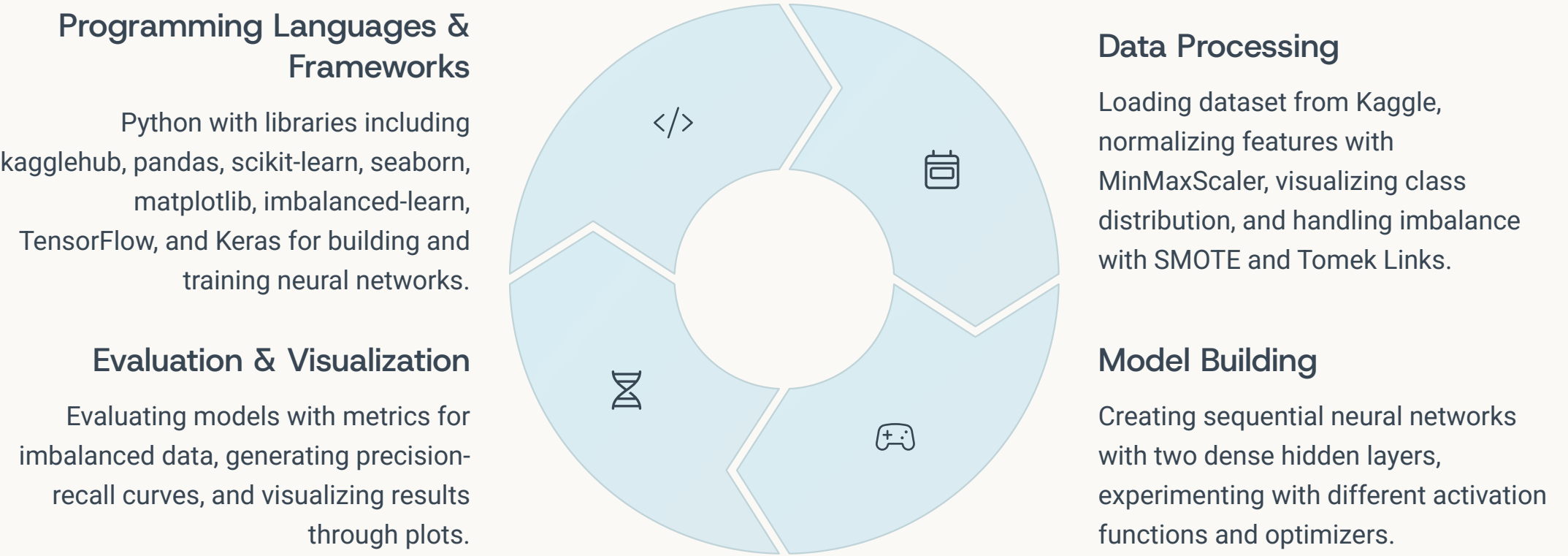
An adaptive learning rate algorithm that combines benefits from AdaGrad and RMSProp. It's generally efficient and widely used.



RMSprop (Root Mean Square Propagation)

Another adaptive learning rate algorithm that adapts learning rates for each parameter.

Implementation Details



The system works as a sequential pipeline, processing the credit card transaction data through several stages:

Data Acquisition & Preprocessing

The process begins by downloading the raw dataset from Kaggle and loading it into memory as a pandas DataFrame. The loaded data is subjected to feature normalization, ensuring that features like 'Amount' are scaled consistently with the PCA-transformed features (V1-V28).

Exploratory Analysis & Imbalance Handling

The normalized data's class distribution is visualised to highlight the extreme imbalance between genuine and fraudulent transactions. To mitigate this bias, the system applies a resampling strategy combining SMOTE and Tomek Links, resulting in a more balanced dataset for training.

Model Training & Experimentation

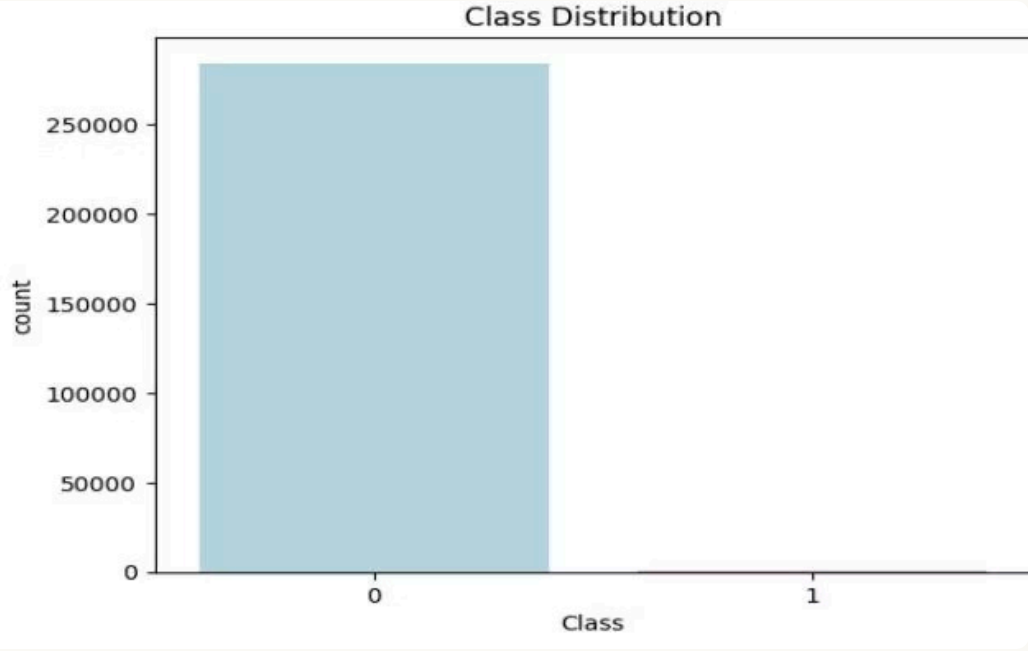
The system builds and trains multiple neural network models, iterating through different combinations of hidden layer activation functions ('relu', 'tanh') and optimizers ('SGD', 'Adam', 'RMSprop'). For each combination, a model is created, compiled, and trained on the training data.

Performance Evaluation & Visualization

After training each model, the system makes predictions on the test set and evaluates performance using metrics like accuracy, recall, and F1-score. Precision-Recall Curves are generated, and the Area Under the Precision-Recall Curve (AUPRC) is calculated for each model configuration.

The code modules are organized to handle each stage of this pipeline, from data loading and preprocessing to model training and evaluation. The results are aggregated and visualized to compare the performance of different model configurations.

Results and Discussion



Evaluation Metrics

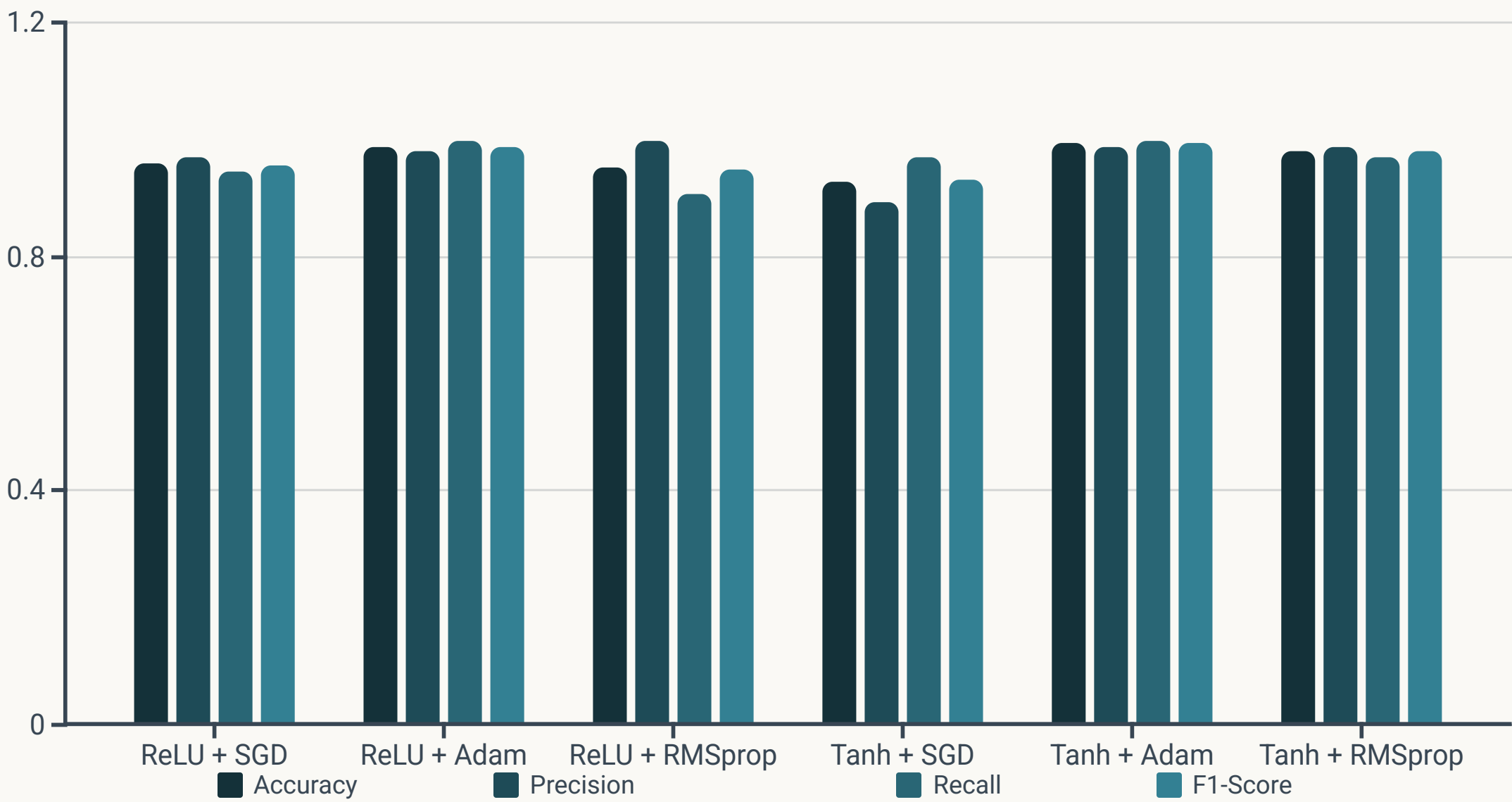
The project uses several key metrics to evaluate model performance:

- Accuracy: The ratio of correctly classified instances to the total number of instances.
- Precision: The proportion of positive predictions that were actually correct.
- Recall: The proportion of actual positive cases that were correctly identified.
- F1 Score: The harmonic mean of precision and recall.
- Area Under the Precision-Recall Curve (AUPRC): Summarizes the trade-off between precision and recall across various thresholds.

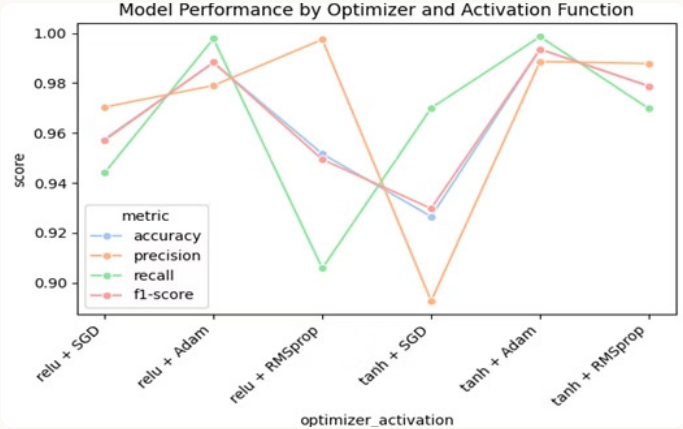
Key Findings

The results show that:

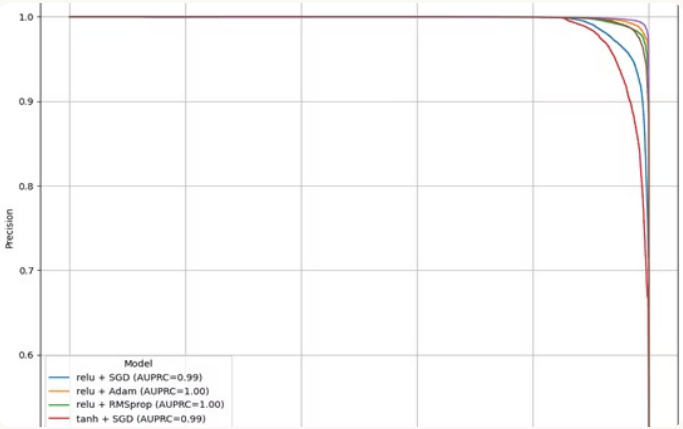
- Models using the Adam optimizer consistently perform best across all metrics.
- The combination of Adam optimizer with tanh activation achieved the highest accuracy (0.993572) and F1-score (0.993617).
- SGD optimizer showed the most unstable learning patterns with erratic validation loss and accuracy fluctuations.
- Most models achieved near-perfect AUPRC scores close to 1.00, indicating strong performance despite the class imbalance.



The visualization of model performance across different configurations reveals several important patterns:



activation	optimizer	accuracy
relu	SGD	0.957556
relu	Adam	0.988209
relu	RMSprop	0.951656
tanh	SGD	0.926543
tanh	Adam	0.993572
tanh	RMSprop	0.978888



Adam + Tanh Performance

The model using the Adam optimizer with tanh activation shows strong generalization, with validation loss staying close to training loss and accuracy steadily improving. This suggests effective learning and minimal overfitting compared to other configurations.

SGD Performance Issues

Models using the SGD optimizer learn inconsistently, with unstable validation loss and accuracy showing sharp fluctuations. This suggests difficulty in finding stable, generalizable patterns compared to Adam and RMSprop.

Precision-Recall Curves

The PR curves compare models using different activation functions and optimizers, showing the trade-off between precision and recall. Higher curves and AUPRC values indicate strong performance, with most models achieving near-perfect scores close to 1.00.

The classification report highlights strong model performance, with an overall accuracy of 0.98. Precision, recall, and F1-scores are consistently high across both classes, indicating balanced and effective classification despite the original class imbalance. This demonstrates the effectiveness of the SMOTE and Tomek Links approach for handling imbalanced data.

Output and misclassified dataset

Fraud Detection Results (First 20 Transactions):

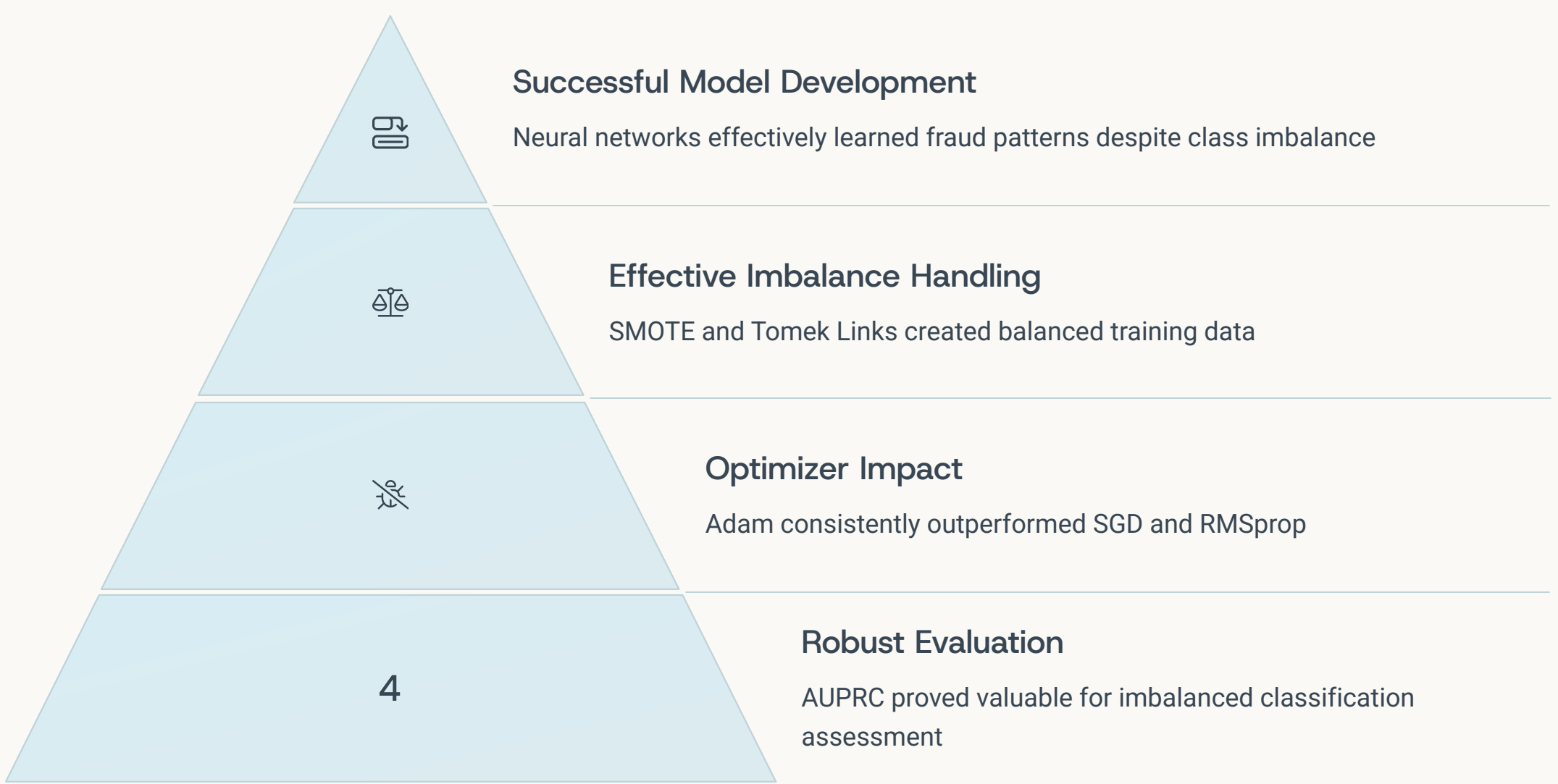
	Amount	Time	y_true	y_pred_prob	y_pred_class
437378	0.005159	0.795437	1.0	0.999911	1
504222	0.002621	0.346162	1.0	0.999948	1
4794	0.001188	0.024694	0.0	0.000813	0
388411	0.000515	0.207959	1.0	0.999999	1
424512	0.000134	0.301639	1.0	0.999885	1
123536	0.000422	0.445443	0.0	0.001805	0
333319	0.002496	0.238206	1.0	1.000000	1
369666	0.006450	0.914337	1.0	0.286808	1
62882	0.000794	0.292056	0.0	0.000047	0
414847	0.000346	0.562204	1.0	1.000000	1
37898	0.008680	0.226631	0.0	0.148034	0
443820	0.006984	0.314959	1.0	1.000000	1
382863	0.003892	0.157227	1.0	0.999984	1
28344	0.000467	0.202347	0.0	0.000090	0
35303	0.001557	0.220236	0.0	0.000417	0
174089	0.000801	0.704674	0.0	0.000172	0
385153	0.001509	0.202305	1.0	0.999999	1
464790	0.009207	0.543156	1.0	1.000000	1
82367	0.017125	0.343615	0.0	0.003154	0
229226	0.002219	0.844049	0.0	0.016373	0

Misclassified Transactions:

	Amount	Time	y_true	y_pred_prob	y_pred_class
229226	0.002219	0.844049	0.0	0.543379	1
187696	0.002975	0.738738	0.0	0.933696	1
207386	0.000584	0.790829	0.0	0.889166	1
238732	0.000039	0.866915	0.0	0.706867	1
64136	0.000030	0.295401	0.0	0.850424	1
...
11589	0.000296	0.115225	0.0	0.944807	1
13890	0.007382	0.142669	0.0	0.565965	1
239846	0.000039	0.869815	0.0	0.936336	1
221831	0.000393	0.825958	0.0	0.843561	1
156841	0.000306	0.630972	0.0	0.844549	1

[2009 rows x 5 columns]

Conclusion & Future Work



This project successfully built and evaluated neural network models for credit card fraud detection using a dataset of European cardholder transactions. The models were trained on normalized features and addressed the significant class imbalance (0.172% fraud cases) using SMOTE and Tomek Links techniques. The neural networks consisted of two dense hidden layers with different activation functions (ReLU, Tanh) and were optimized using various algorithms (SGD, Adam, RMSprop).

Performance evaluation using metrics suitable for imbalanced datasets showed that models with the Adam optimizer consistently performed best, with the Tanh+Adam combination achieving the highest accuracy (0.993572) and F1-score (0.993617). The Area Under the Precision-Recall Curve (AUPRC) served as a robust metric for comparing model performance on the imbalanced dataset.

Limitations

- Limited Dataset Temporal Coverage: Only two days of transaction data, missing seasonal trends and evolving fraud patterns
- Fixed Model Architecture: Relatively simple architecture with predefined structure
- Restricted Hyperparameter Search: Limited exploration of activation functions and optimizers
- Specific Resampling Approach: Potential drawbacks of SMOTE and Tomek Links
- Fixed Classification Threshold: Using 0.5 threshold without optimization

Future Improvements

- Utilize More Extensive Data: Train on larger datasets spanning longer periods
- Advanced Model Architectures: Experiment with more complex neural networks
- Comprehensive Hyperparameter Optimization: Implement systematic tuning techniques
- Explore Alternative Imbalance Handling: Investigate other techniques
- Optimize Classification Threshold: Determine optimal threshold based on business needs
- Feature Engineering: Create new features from existing data
- Regularization Techniques: Implement dropout and L1/L2 regularization
- Cross-Validation: Employ k-fold cross-validation for robust assessment

The project demonstrates the effectiveness of neural networks for fraud detection when properly addressing class imbalance and selecting appropriate activation functions and optimizers. Future work could expand on these foundations to create even more robust and adaptable fraud detection systems.

References

20

References Cited

Academic sources supporting this research

6

Activation & Optimizer

Combinations tested in neural networks

0.172%

Fraud Transactions

Percentage in the original dataset

0.993

Best F1-Score

Achieved with Tanh+Adam configuration