A HYBRIDIZED SMOTE-ENN APPROACH ON IMBALANCED DATASET OF FRAUDULENT CREDIT-CARD SCENARIO

By

ENESI FEMI AMINU * ABDULQADRI OLALEKAN ARAOYE ** AYOBAMI EKUNDAYO ***

OLUWASEUN ADENIYI OJERINDE **** GRACE AMINA ONYEABOR *****

*-**** Department of Computer Science, Federal University of Technology, Minna, Nigeria.
***** Department of Information Technology, Federal University of Technology, Minna, Nigeria.

https://doi.org/10.26634/jds.3.1.21583

Date Received: 14/01/2025 Date Revised: 07/02/2025 Date Accepted: 27/03/2025

ABSTRACT

Businesses and financial activities are now carried out effortlessly thanks to the advancement of information technology. Credit cards make it simple and comfortable to perform company activities remotely. However, this advancement is not without obstacles and compromises, since credit card fraud is expanding at an exponential rate. Thus, in order to address this difficulty using cutting-edge deep learning technology to detect fraud, the dataset in question must be easily available and balanced. However, most of the available datasets are not balanced, thereby potentially affecting the accuracy of the learning models to detect or classify. To this end, this study aims to hybridize the Synthetic Minority Oversampling Technique-Edited Nearest Neighbor (SMOTE-ENN) algorithm to balance the dataset and detect the possibility of fraud. SMOTE is taken into consideration in order to proffer a solution to the imbalanced nature of the dataset, which was acquired from the Kaggle repository based on the insight of the benchmark literature. The ENN, which is the deep neural network, would in turn receive the output from this process. Based on the results, the hybridized technique is promising because the model was able to record an accuracy and F1-score of 99%.

Keywords: Credit Fraud, Data Imbalance, SMOTE ENN, Fraud Detection, Machine Learning.

INTRODUCTION

Credit cards rapidly replaced cash because they were more convenient in previous decades as businesses adapted to the digital world and financial operations moved toward computerized administration inside an expanding cashless banking environment. Credit cards thereafter became the most common way to make payments (Mishra & Ghorpade, 2018). There are two main types of credit card-based transactions: physical card purchases and virtual card purchases. The latter, which primarily pertains to purchases conducted online, only needs basic data such as card numbers, expiration dates, and security codes. The number of people using credit cards increased along with the amount of fraudulent transactions (Shamsudin et al., 2020). Credit card fraud refers to situations in which the sensitive data linked to the credit card is compromised or the actual credit card is misplaced. This may entail using the card fraudulently and without authorization (Hordri et al., 2018). The increasing sophistication of fraudulent operations makes credit card fraud detection a key issue for financial institutions (Adebayo et al., 2023).

Financial institutions and enterprises can improve their capacity to promptly and effectively detect suspicious actions by utilizing these sophisticated computational tools. This will contribute to the protection of cardholders

and the financial ecosystem as a whole (Shamsudin et al., 2020). It is possible to identify certain types of fraudulent activity by using different machine-learning algorithms. These algorithms are intended to find possible fraud cases by examining trends, abnormalities, and discrepancies in transaction data. Predictive models are essential to supervised fraud detection techniques in the field of credit card fraud detection. These models are created by using samples of both authentic and fraudulent transactions as training data. The goal is to empower these models to accurately classify newly entered transactions, classifying them as either valid or fraudulent. However, credit card fraud falls within the category of highly unbalanced datasets that are available to the public (Mishra & Ghorpade, 2018). Consequently, if necessary strategies were not employed to deal with this challenge, it would inadvertently affect the expected accuracy of the result obtained from the models employed for prediction. Based on literature, some of these strategies to balance the given datasets have been deployed; for example, Synthetic Minority Oversampling Techniques (SMOTE).

The research of Aftab et al. (2023) and Gupta et al. (2023) employed SMOTE to correct the imbalanced nature of the credit card fraud detection dataset. The SMOTE-ENN is the greatest option for quick fraud detection because testing has shown that it is more sensitive and accurate than other machine learning techniques. Compared to models like MLP, LSTM, GRU, AdaBoost, and Random Forest (Medida et al., 2024). Also, in the study of Khalid et al. (2024), it was stated that a comparative study of the suggested ensemble model, conventional machine learning techniques, and individual classifiers shows that the ensemble models perform better at reducing the difficulties related to credit card fraud detection. Regarding recall, accuracy, precision, and F1-score.

Muntasir Nishat et al. (2022) canvassed for a hybridized approach that can combine undersampling of the minority class with oversampling in order to address imbalances in the dataset. This is because, as earlier reported, the use of machine learning models (ML) for fraud detection is significantly hampered by the presence of datasets that are wildly out of balance (Dhiman et al., 2023). Based on literature, several supervised machine learning models have been constantly employed for classification or prediction purposes. For example, the study of Afriyie et al. (2023) studied the performance of three different machine learning models (logistic regression, random forest, and decision trees) to classify, predict, and detect fraudulent credit card transactions. However, for better prediction accuracy, deep learning models are promising, especially for large-volume datasets. Models such as decision trees, random forests, support vector machines, logistic regression, and artificial neural networks have demonstrated their significance over time. Therefore, this research work aims to hybridize the SMOTE and ENN techniques in order to have better accuracy from the proposed model, which is the deep neural network. The SMOTE-ENN strategy under-samples the dataset's redundancies using the ENN approach after oversampling the dataset's minority class using interpolation. The dataset used in this research is obtained from the Kaggle repository, and it was downloaded in a comma-separated value (CSV) format.

1. Related Works

The market economic order has been negatively harmed by credit card theft, which has also damaged financial institutions, stakeholders, and customers' confidence. The unbalanced dataset associated with credit card fraud transactions outweighs the magnitude of fraud being committed. The unbalanced data issue, which arises when one class's examples far exceed those of the other class prior to resolving the fraud issue. This unbalanced dataset makes it difficult to classify fraud because the results could be skewed in favor of the dominant group (Abd El-Naby et al., 2023).

According to Sasank et al. (2019), the use of credit cards for online purchases has significantly increased as a result of e-commerce. Regrettably, there has been an increase in credit card fraud as a result of this spike in credit card usage. These incidents may be rare, yet they can have a significant impact. With various machine learning approaches, facilitators have been actively attempting

to improve credit card transaction fraud detection. Nonetheless, the imbalance in the dataset is one of the fundamental issues this domain faces. This paper utilized five machine learning approaches, such as Random Forest, Naive Bayes, Support Vector Machine, K-Nearest Neighbor, and Logistic Regression. Various sampling strategies, including oversampling, undersampling, both sampling, random oversampling examples (ROSE), and synthetic minority over-sampling technique (SMOTE), are applied to the balanced dataset. As demonstrated by the research work's remarkable accuracy rate of 97.04% and precision rate of 99.99%, logistic regression stands out.

Credit card fraud (CCF) is a kind of fraud wherein somebody other than the record holder utilizes a charge card or record data for an unapproved exchange. In lieu of this, Sadineni (2020) aimed to detect fraudulent transactions using credit cards with the help of machine learning algorithms and deep learning algorithms. The machine learning models utilized in this study are SVM, RF, KNN, DT, LR, Voting Classifier, XGBoost, MLP, Standard BL, CNN + LSTM, and CNN. The result among ideal qualities for exactness, F1 score, accuracy, and AUC bends about 99.9%, 85.71%, 93%, and 98%, respectively.

Along with the frightening rise in internet shopping, there has also been a worrying rise in fraudulent transactions. Ahammad et al. (2020) aimed at detecting credit card fraud using data pre-processing on imbalanced data with both oversampling and undersampling. Data pre-processing was done through data cleaning and data integration. The machine learning model used in this study is the K-Nearest Neighbor. Accuracies obtained without sampling, with undersampling only, with oversampling, and with both undersampling and oversampling are 98.01%, 72.13%, 76.89%, and 81.21%, respectively.

Sisodia et al. (2017) emphasized that banks and other financial institutions now urgently require reliable fraud detection systems. The problem of class imbalance becomes apparent when one considers how seldom fraudulent transactions are in comparison to legitimate ones. Using the dataset, the authors apply SMOTE, SMOTE ENN, SAFE SMOTE, ROS, and SMOTE TL. These methods aim to address the imbalance problem. The resampled data is then evaluated using two types of classifiers: ensemble classifiers (Adaboost, Bagging) and cost-sensitive classifiers (CSVM, C4.5). Key performance parameters, including sensitivity, specificity, G-mean, and area under the ROC, are used to assess these classifiers' effectiveness. The findings indicate that the SMOTE ENN strategy outperforms the other oversampling techniques examined in terms of fraud instance detection.

There has been a significant increase in fraud incidents as a result of both the growing number of businesses and customers using credit cards to complete financial transactions. Alenzi and Aljehane (2020) aimed at detecting frauds in credit cards using logistic regression. Data pre-processing in this study is done using two novel main methods to clean the data: the mean-based method and the clustering-based method. The machine learning model used in this study is the logistic regression. The accuracy, sensitivity, and error rate metrics of the proposed logistic regression-based classifier are assessed. It is compared with two popular classifiers: the voting classifier and the K-nearest-neighbors classifier. The logistic regression-based classifier produces the best results (accuracy = 97.2%, sensitivity = 97%, and error rate = 2.8%).

Shamsudin et al. (2020) highlighted that credit card theft is a serious issue exacerbated by the extremely unbalanced distribution of classes. The scientists recommended taking care of this issue in the preprocessing stage. In order to address imbalanced data, sampling techniques, especially under- and oversampling, are frequently employed. The study employs a range of oversampling techniques combined with random undersampling; the performance assessment shows significant improvements in important metrics like F1-score, precision, and recall. These gains, on average, result in an impressive F1-score value of 0.80%, demonstrating the effectiveness of the suggested strategy.

The use of digital currency has grown around the world, and with it has come an increase in fraud. Azhan and

Meraj (2020) aimed to detect credit card fraud using machine learning and deep learning techniques. The machine learning algorithms used are Multinomial Naive Bayes, Random Forest Regression, Logistic Regression, Support Vector Machine, and a basic Neural Network. KNN and NN had 0.75 and 0.78 F1 - score and precision, respectively.

Udeze et al. (2022) aimed at using machine learning and resampling techniques for credit card fraud detection. Three machine learning models were used in this study (RF, XGBoost, and TensorFlow Deep Neural Network). The DNN is more efficient than the other two algorithms in modeling the undersampled datasets, while overall, the three algorithms had a better performance in the oversampling than the undersampling technique. The dataset used in this study was obtained from the Kaggle repository. Data pre-processing was done using Principal Component Analysis (PCA). ADASYN from the imbalanced-learn library was used as the data oversampling technique. An accuracy and F1- score of 99% and 87% were obtained, respectively.

Financial institutions and credit card holders suffer large financial losses as a result of credit card fraud. The study of Alarfaj et al. (2022) aimed to detect such frauds, including the accessibility of public data, high-class imbalance data, the changes in fraud nature, and high rates of false alarm. The feature selection technique used in this study is the principal component analysis (PCA). The machine learning and ensemble learning techniques used in this study include Extreme Learning Method, Decision Tree, K-Nearest Neighbor, Random Forest, Support Vector Machine, Logistic Regression, and XGBoost. Also, the baseline model and CNN were the deep learning techniques used in this study. The evaluation of this study shows the improved results achieved, such as accuracy, F1- score, precision, and AUC curves having optimized values of 99.9%, 85.71%, 93%, and 98%, respectively.

Fraud is a major problem when it comes to utilizing credit cards, especially in the context of online transactions. As a result, Alfaiz and Fati (2022) aimed to enhance a credit card fraud detection model utilizing a machine learning strategy. The dataset used in this study was obtained from the Kaggle repository. The nine machine learning models used in the first stage of the proposed approach are LR, KNN, DT, NB, RF, GBM, LightGBM, XGBoost, and CatBoost. The results indicate that the proposed model outperforms previous models with an AUC value of 97.94%, a recall value of 95.91%, and an F1-score value of 87.40%.

The number of sales and purchases made online is growing every day, and most of them include credit card transactions. The end-user benefits from this in addition to the fact that it makes credit card theft online more common. Alharbi et al. (2022) aimed to develop a novel text2IMG mechanism for credit card fraud detection. This study obtained a dataset from the Kaggle repository, which was used to develop a deep learning (DL)-based approach to solve the text data problem. Three variants of the nearest-neighbor methods, such as fine-KNN, medium-KNN, and coarse-KNN, and three variants of ensemble methods, such as LPBoost, bagged-boost, and subspace ensemble boost, are used. An accuracy of 99.87% was achieved by Coarse-KNN using deep features of the proposed CNN.

Advances in electronic commerce and communication networks in recent years have led to a large growth in the use of credit cards for conventional and online transactions. Nevertheless, there has been a constant increase in credit card fraud, which costs financial organizations enormous sums of money annually. In lieu of this, Esenogho et al. (2022) proposed an efficient approach to detect credit card fraud using a neural network ensemble classifier and a hybrid data resampling method. The ensemble classifier is obtained using a long short-term memory (LSTM) neural network as the base learner in the adaptive boosting (AdaBoost) technique. Also, the hybrid resampling is achieved using the synthetic minority oversampling technique and edited nearest neighbor (SMOTE-ENN) method. The experimental findings demonstrate that the classifiers outperformed the other methods when trained on the resampled data. The suggested LSTM ensemble achieved a sensitivity and specificity of 0.996 and 0.998, respectively, outperforming the other techniques.

2. Methodology: Hybridization Strategy for SMOTE-ENN Algorithm

This research proposed the hybridization approach for the SMOTE-ENN algorithm in order to obtain better results accuracy on the imbalanced credit card dataset and ultimately leads to robust pattern identification owing to the proposed deep neural network model. Figure 1 shows an overall conceptual framework for the proposed system.

Based on the conceptual framework, it can be classified into three tiers, namely the first tier, which is the data collection layer; the second tier is the data preprocessing layer; and lastly, the data training layer.

In the first tier of the proposed methodology, the credit card fraud raw dataset is obtained from the Kaggle repository, and it was downloaded in a comma-separated value (CSV) format. The data scientist loaded this dataset into the Jupyter Notebook of the Python programming language. The data collected contain 284807 data points (rows) and 31 features (columns) describing the independent variable and non-independent variable. The independent variable describes various features or characteristics of the credit card transaction variable, while the dependent variable determines if the transaction is fraudulent or not. The second tier of the proposed conceptual framework is the data preprocessing stage, which comprises the data exploration, data cleaning, and data balancing stages. The practice of looking over, evaluating, and displaying data in order to comprehend its attributes, patterns, relationships, and structure is known as data exploration. The data exploration involves correlation analysis, clustering, dimensionality reduction, and other statistical techniques. Data cleaning involves inputting missing values, removing outliers, standardizing formats, and correcting errors. Lastly, balancing of the raw credit card fraud dataset obtained from the Kaggle repository is done by employing the Synthetic Minority Oversampling Technique and Edited Nearest Neighbor (SMOTE-ENN).

After the dataset has been balanced by making sure that

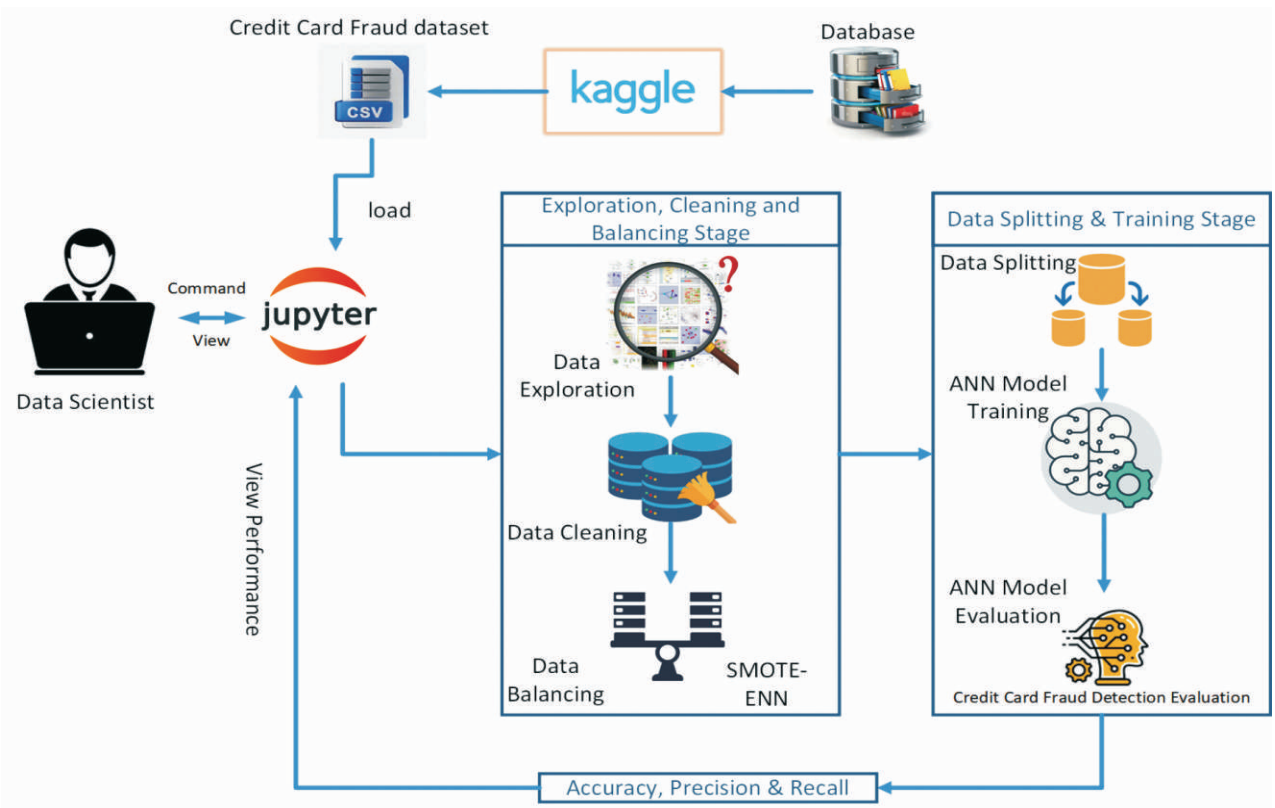

Figure 1. Conceptual Framework for the Proposed Credit Card Fraud Detection System

the majority class is equal to the minority, the data splitting and data training stage is carried out. In order for the classification model to classify the dataset appropriately, the dataset must be balanced. The third tier is composed of the model classification process that uses the ANN based on the diagram, and also, the created model is evaluated. As shown in the conceptual model, the data scientist is the entity carrying out each of these different steps. The algorithmic design for the proposed model is given below.

2.1 Component Details and Algorithm

Input: CSV base Credit Card Fraud Dataset

Output: Detection Result (d)

Parameters: credit card fraud dataset (t),Jupyter notebook (j), Data exploration (de), Data cleaning (dc), clean data (c_t), Data balancing (db),SMOTE-ENN, Data spiltting (ds), oversampling (o_s), undersampling (u_s), imbalanced dataset (imb), balanced dataset (b_d),

Procedure:

Input t

c_t = Preprocess t (de, dc)

b_d = SMOTE(c_t)

For Each b_d

Stop o_s And u_s

End For

Splitting b_d

Invoke SMOTE-ENN (b_d)

Output d

The obtained dataset from Kaggle is inputted for preprocessing such that data cleaning and exploration were carried out, and the output of this process is denoted as clean data, as shown by lines 1 to 2. Because of the imbalanced nature of the dataset, the SMOTE technique is employed, as shown by line 3, whose input is the clean data and is expected to produce balanced data. The method must ensure that for data to be balanced, there must be no form of oversampling or undersampling. From lines 7 to 9, the balanced data is split into training and testing. The training portion is fed into the hybridized SMOTE-ENN, and the result is produced based on the metrics of accuracy, precision, recall, and F1 - score. Figure 2 shows the sequence diagram of the proposed study.

The sequence diagram also shows entity and functionality at each instance of any operation. It shows the user checking for fraudulent activity by making a request call through the user interface, which further initializes the model by importing the prediction model parameter, and then the credit card prediction model is loaded through a local file. Also, the prediction model makes predictions, and the prediction outcome is returned as binary output, which is viewed by the end user.

3. Results and Discussion

The suggested credit card fraud detection model was developed using Python version 3.9 and a few key Python modules, including Numpy, Pandas, Tensorflow, Sklearn, and Mathplotlib. The Jupyter Notebook Lab is the integrated development environment (IDE) that is taken into consideration for writing the complete model development process. A minimum of 4GB of RAM and a CPU speed of 2.0 GHz are required for constructing the suggested model's hardware. A TPU and GPU processor may also be used.

3.1 Model Implementation

The descriptive phase of creating the credit card prediction model is covered in this section. It includes the importing of data and libraries and the investigation, analysis, visualization, cleaning, balancing of data, model training, and evaluation. Figure 3 shows the library importation and dataset loading.

The necessary library imported for the development and execution of different methodological steps for the credit card detection model is shown. The libraries that were imported are: Tensorflow, sklearn, and Keras for data preprocessing, model training, and evaluation; NumPy is needed for numerical computation; Matplotlib is needed for data visualization; and pandas is needed for data import, updating, and indexing. Additionally, the READ_CSV command is used to import the credit card dataset into the Jupyter Lab IDE environment. The PANDAS head method is used to visualize five sample data points,

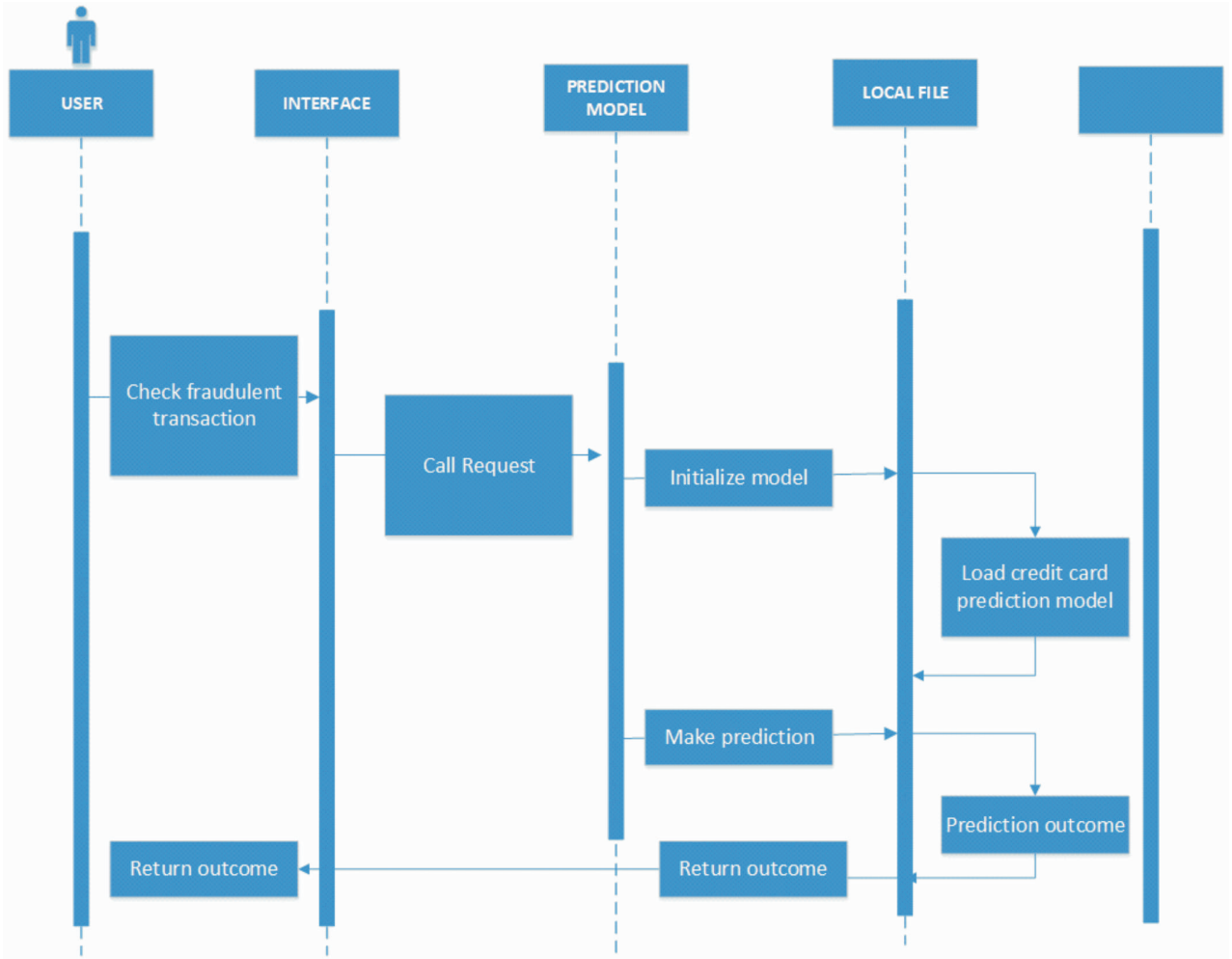

Figure 2. Credit Card Fraud Sequence Diagram

and the sample data indicates that the majority of the data points are floating-point numbers. After the importation of the various libraries needed. The next stage is the data exploration stage, which displays the number of rows and columns in the credit card fraud detection dataset.

Figure 4 shows the data exploration. It shows the total number of rows and columns in the credit card fraud detection data set. There are thirty-one (31) features, including independent and dependent variables, and two hundred and eighty-four thousand eight hundred and seven (284,807) data points or rows in the dataset. It also displays the pandas module using the INFO function.

Figure 5 shows that every entry in the dataset is complete and has no missing or empty values. The three key details shown are based on the feature names, the count of non-null (non-empty records), and the data type of the values in each column.

The dataset's count of empty values in each column is displayed in Figure 6.

The is_null function is used to retrieve this data, and the sum is then accumulated. There are no empty records in any of the columns indicated by the zero values for any column. The code snippet for obtaining duplicate records in the dataset is displayed in Figure 7.

Duplicate data compromises the dataset's integrity; the figure indicates that 1,081 sample records are duplicates. Therefore, it is imperative that the duplicate value be eliminated. The duplicate value and the total of the duplicates that are gathered using duplicated() are

importing library

```
[148]: import pandas as pd
       import matplotlib.pyplot as plt
       from sklearn.preprocessing import MinMaxScaler
       from sklearn import model_selection
       import warnings
       import tensorflow as tf
       import keras
       import numpy as np
       from sklearn import metrics
       import seaborn as sn

       warnings.filterwarnings("ignore")
```

Loading dataset

```
[149]: # read credit card dataset
       data  = pd.read_csv("dataset/creditcard.csv")
       data.head()
```

[149]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...

5 rows × 31 columns

Figure 3. Library Importation and Dataset Loading

Data exploration

```
[6]: rows, col = data.shape
     print("Rows Count : {}          Column Count : {}".format(rows, col))
```

```
Rows Count : 284807          Column Count : 31
```

```
[152]: data.columns
```

```
[152]: Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
              'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20'
              'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
              'Class'],
             dtype='object')
```

Figure 4. Data Exploration

```
[10]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
```

Figure 5. Credit Card Feature Information

```
[11]: data.isnull().sum()

[11]: Time      0
      V1        0
      V2        0
      V3        0
      V4        0
      V5        0
      V6        0
      V7        0
      V8        0
      V9        0
      V10       0
      V11       0
      V12       0
      V13       0
      V14       0
      V15       0
      V16       0
      V17       0
      V18       0
      V19       0
      V20       0
      V21       0
      V22       0
      V23       0
      V24       0
      V25       0
      V26       0
      V27       0
      V28       0
      Amount    0
      Class     0
      dtype: int64
```

Figure 6. Validating Null Value in Each Column

what the duplicated method will return sum. Figure 8 shows the number of sample points in each output variable class.

The output variable has two unique classes, the non-fraudulent (0) and fraudulent (1) classes, according to the investigation that was done. A small sample of 492 data points is found in the fraudulent class, compared to a sample of 284,315 data points in the non-fraudulent transaction. This indicates a significant degree of data imbalance, which may cause biased predictions.

3.2 Credit Card Data Preprocessing

A step-by-step breakdown of the data-cleaning procedure is covered in this section. The earlier dataset investigation serves as the foundation for each cleaning procedure. The credit card dataset must have the duplicate data sample dropped or removed due to the existence of duplicate data entry. The scaling procedure will be triggered by a high degree of distribution in time and quantity of data points. Additionally, the sample data will be divided into testing and training samples. The code snippet used to eliminate duplicate data entries from the credit card dataset is shown in Figure 9.

The drop_duplicates() method is used to provide this feature; when duplicate entries are removed, the total data sample decreases from 284,807 samples to 283,726 samples. The transformation procedure on the highly scattered columns Time and Amount is shown in Figure 10.

```
[12]: duplicate = data.duplicated().sum()
      print("Numbers of Duplicate: {}".format(duplicate))

      Numbers of Duplicate: 1081

[13]: print('Duplicate Samples.....')
      print("==========================")
      data[data.duplicated]

      Duplicate Samples.....
      ==========================
```

[13]:

	Time	V1	V2	V3	V4	V5	V6	
33	26.0	-0.529912	0.873892	1.347247	0.145457	0.414209	0.100223	0.711
35	26.0	-0.535388	0.865268	1.351076	0.147575	0.433680	0.086983	0.693
113	74.0	1.038370	0.127486	0.184456	1.109950	0.441699	0.945283	-0.036
114	74.0	1.038370	0.127486	0.184456	1.109950	0.441699	0.945283	-0.036
115	74.0	1.038370	0.127486	0.184456	1.109950	0.441699	0.945283	-0.036
...	...	...	...	...	...	...	...	
282987	171288.0	1.912550	-0.455240	-1.750654	0.454324	2.089130	4.160019	-0.881
283483	171627.0	-1.464380	1.368119	0.815992	-0.601282	-0.689115	-0.487154	-0.303
283485	171627.0	-1.457978	1.378203	0.811515	-0.603760	-0.711883	-0.471672	-0.282
284191	172233.0	-2.667936	3.160505	-3.355984	1.007845	-0.377397	-0.109730	-0.667

Figure 7. Dataset Duplicate Value

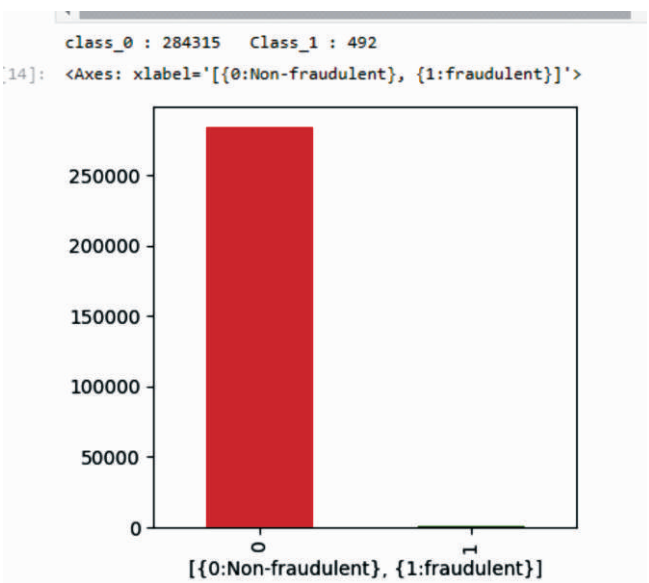

Figure 8. Class Distribution Count

The Min-max Scaling module is used to accomplish the scaling transformation process. Fit_transform() must be called, and the MinMaxScaller() object must be created. Reattached to the primary dataset is the transformed data. To have distinct data for machine learning training and the remaining data for assessing the machine learning model's performance, data splitting is necessary.

3.3 Credit Card Dataset Splitting

The dataset is divided into X_train, X_test, y_train, and y_test by calling the method train_text_split() using the model_selection module. The input feature size, 226,980, and 56,747 data samples are the X train and test. The Y train and test are subject to the same sample points. The real dataset used in this study is split up into 20% testing and 80% training samples. Figure 11 shows the process of splitting the card dataset into training and testing sets.

Data Cleanning

```
[153]: # removing dublicate
clean_data = data.drop_duplicates()
clean_data
```

[153]:

	Time	V1	V2	V3	V4	V5	V6	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.
...	...	...	...	...	...	...	...	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.

283726 rows × 31 columns

Figure 9. Dropping or Removal of Duplicate Sample

```
[156]: # Normalize highly distributed Data..... (Time and Amount)
min_max = MinMaxScaler()
min_max
```

```
[156]: MinMaxScaler()
```

```
[157]: time = min_max.fit_transform(clean_data[['Time']])
amt = min_max.fit_transform(clean_data[['Amount']])
```

```
[158]: clean_data.Time  = time
clean_data.Amount = amt
```

```
[159]: clean_data[["Time", "Amount"]]
```

[159]:

	Time	Amount
0	0.000000	0.005824
1	0.000000	0.000105
2	0.000006	0.014739
3	0.000006	0.004807
4	0.000012	0.002724

Figure 10. Time and Amount Data Scaling

```
i2]: X_train, X_test, y_train, y_test = model_selection.train_test_spl

print("X train :{}   \nX test :{}   \ny train :{}   \ny test:{}".
```

```
X train :(226980, 30)
X test :(56746, 30)
y train :(226980,)
y test:(56746,)
```

Figure 11. Card Dataset Splitting

3.4 Data Balancing (SMOTE-ENN)

This strategy is used to correct the credit card dataset's imbalanced class distribution. Figure 12 shows the application of SMOTEENN to resample the dataset, resulting in balanced class distributions for further analysis.

3.5 SMOTE-ENN Implementation

The pre-implementation functionality of SMOTE-ENN is available to programmers through the imblearn combine Python module. The fit_sample method is used after

```
[167]: from imblearn.combine import SMOTEENN

[172]: smote_enn = SMOTEENN(random_state=42)
       X_res, y_res = smote_enn.fit_resample(X, y)

[174]: print("X sample   :{} |   y   sample :{}".format(X.shape, y.shape))
       print("X resmaple :{}     y resample :{}".format(X_res.shape, y_res.shape))

       X sample   :(283726, 30) y   sample :(283726,)
       X resmaple :(566098, 30) y resample :(566098,)

[177]: X_res.to_csv("X_resampled")
       y_res.to_csv("y_resampled")
```

Figure 12. SMOTE-ENN Resampling of the Dataset

creating an instance of SMOTEENN in order to resample the majority class using SMOTE and reduce noise using ENN methods. The parameters for the fit_resample technique are y, the dependent or output feature, and X, the independent feature. The bar chart in Figure 13 shows the balanced data of equal class that was produced as a result of the SMOTE-ENN data balancing technique. It visualizes the value count for each class. Each class has

```
print(y.Class.value_counts())
print("==============================")
plt.figure(figsize=(4,4))
y.Class.value_counts().plot(kind='bar', color=['b', 'green'])
plt.show()
```

```
1    283253
0    282845
Name: Class, dtype: int64
==============================
```

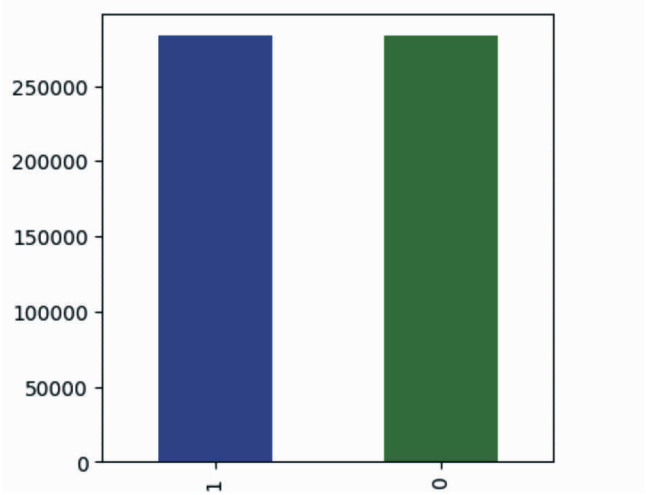

Figure 13. SMOTE-ENN Implementation

an equal number of samples, as evidenced by the equal bar length in the bar chart.

3.6 DNN Model Development

The training process of the deep learning model for credit card prediction takes the deep neural network architecture into account. This section will cover model training, model assessment, and model architecture layers.

The DNN model architecture's layers and model configuration are depicted in Figure 14. The sequential layer of the deep neural network architecture is defined using the TensorFlow framework, as shown in Figure 14. According to the figure, the deep neural network's sequential layer is made up of six (6) layers: one (1) input layer, two (2) dense layers, two (2) dropout layers, and one (1) output layer. In order to eliminate non-linearity in the training weight, each completely linked dense layer is constructed with an activation function of ReLU. The input layer is defined with 30 dimensions, or 30 input features. In order to address the overfitting problems during training, the dropout layer is also introduced. It works by removing 50% of the neurons after the dense layer. One (1) neuron and a sigmoid activation function complete the definition of the output layer.

Moreover, key parameters like optimizer, metrics, and loss must be compiled into the DNN model that was developed to identify fraudulent activity in credit card transactions. Because credit card fraud detection is a

```
[5]: dnn_model = tf.keras.Sequential([
        tf.keras.Input(shape=(30,)),
        tf.keras.layers.Dense(24, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(15, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(1, activation='sigmoid')
    ])

    dnn_model.compile(
        optimizer='adam',
        metrics=['accuracy'],
        loss = "binary_crossentropy"
    )
    dnn_model.summary()

    Model: "sequential_5"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #
    =================================================================
     dense_12 (Dense)            (None, 24)                744

     dropout (Dropout)           (None, 24)                0

     dense_13 (Dense)            (None, 15)                375

     dropout_1 (Dropout)         (None, 15)                0

     dense_14 (Dense)            (None, 1)                 16

    =================================================================
    Total params: 1,135
    Trainable params: 1,135
    Non-trainable params: 0
```

Figure 14. DNN Model Architecture

binary classification job, the Adam optimizer is taken into account, and the binary_crossentropy is employed as the loss metric. The architecture of the deep learning model is depicted in the above picture, and the model's summary() function is used to visualize it.

3.7 DNN Model Training

The fit method of the sequence class is called by the DNN model object in order to train the credit card fraud detection. The fit technique takes into account the training samples for X_train and Y_train and the number of training sessions, which is indicated by the keyword epoch, which has been set to five (5). Based on Figure 15, the accuracy and training loss values are displayed for each training step. By the fifth iteration, or epoch, an accuracy of 97.64% with a loss of 0.0550 is attained. The accuracy and loss progression for each epoch are displayed in Figure 16.

The aforementioned chart depicts each loss and accuracy for each epoch as a line graph. This provides a comprehensive summary of how accuracy increased over the course of five epochs and how loss decreased at each epoch. The credit card fraud detection model's classification report using DNN architecture and SMOTE-ENN data balancing approaches is displayed in Figure 17.

The precision, recall, and F1-score are all 99%, and the overall accuracy attained is 99%. Additionally, every class has 99% accuracy because of the dataset's balanced nature. The number of successfully predicted values against the misclassified or mispredicted output is displayed graphically in Figure 18.

The number of successfully predicted values against the misclassified or mispredicted output is displayed graphically in the confusion matrix. According to Figure 18, fifty-six thousand one hundred and eighty-six (56,186) times the non-fraudulent prediction is accurate, and 341 times there is a miscategorization. On the other hand, 301 misclassifications occur while the model is correctly classified 56,392 times for the fraudulent class. The accuracy comparison of individual class accuracy when

```
]: train_hist = dnn_model.fit(X_train, y_train, epochs=5)

   Epoch 1/5
   14153/14153 [==============================] - 27s 2ms/step - loss: 0.1124 - accuracy: 0.9530
   Epoch 2/5
   14153/14153 [==============================] - 27s 2ms/step - loss: 0.0610 - accuracy: 0.9730
   Epoch 3/5
   14153/14153 [==============================] - 25s 2ms/step - loss: 0.0580 - accuracy: 0.9743
   Epoch 4/5
   14153/14153 [==============================] - 35s 3ms/step - loss: 0.0564 - accuracy: 0.9758
   Epoch 5/5
   14153/14153 [==============================] - 39s 3ms/step - loss: 0.0550 - accuracy: 0.9764
```

Figure 15. DNN Model Training

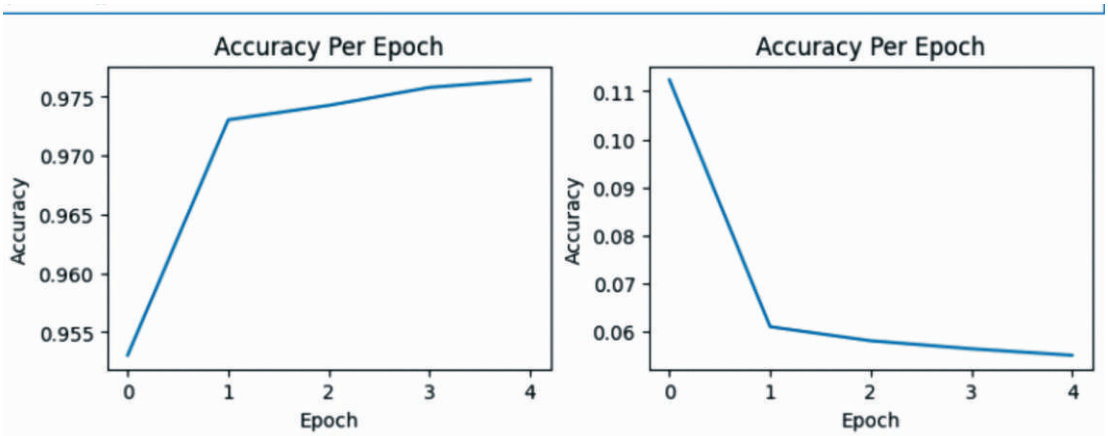

Figure 16. DNN Training Loss and Accuracy

```
0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1]

              Classification Report
    ============================================
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     56527
           1       0.99      0.99      0.99     56693

    accuracy                           0.99    113220
   macro avg       0.99      0.99      0.99    113220
weighted avg       0.99      0.99      0.99    113220

[190]: # report
```

Figure 17. Fraud Credit Card Detection Classification Report

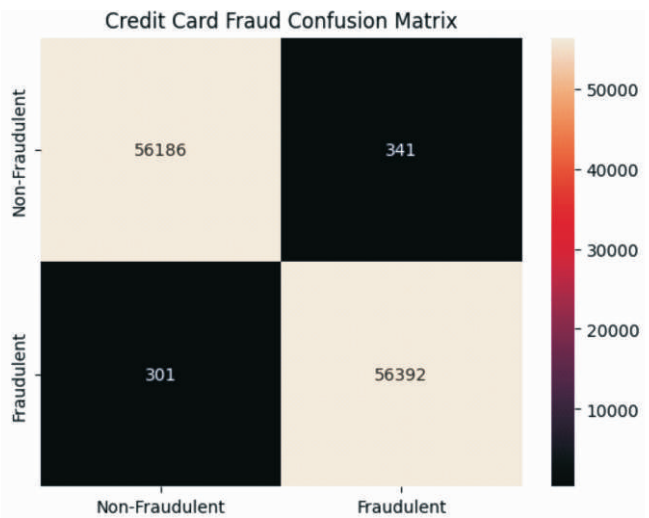

Figure 18. Fraud Credit Card Detection Confusion Matrix

the dataset is unbalanced versus when the model is trained on a balanced dataset using SMOTE-ENN is depicted in Figure 19.

The DNN and SMOTE-ENN dataset sample are used to

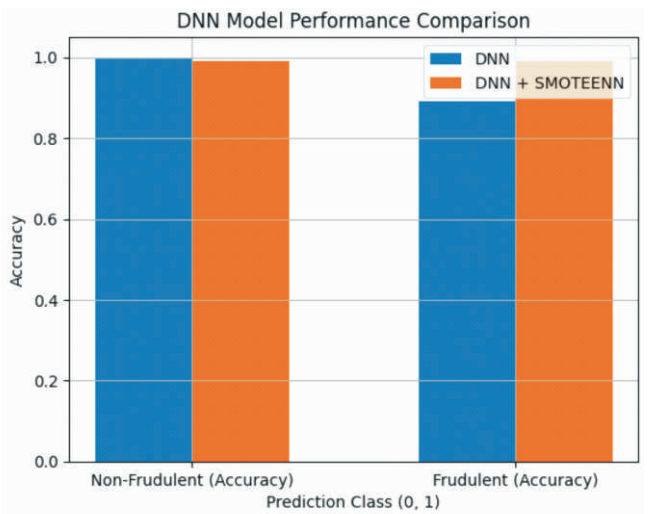

Figure 19. Fraud Credit Card Detection Model Performance Comparison

achieve equal precision (99% accuracy) for each individual class (0: non-fraudulent (99%), 1: fraudulent (99%)), whereas the biased prediction in the imbalanced dataset leads to unequal precision (0: non-fraudulent (100%), 1: fraudulent (89%)).

However, other hybrid resampling techniques, such as SMOTE-Tomek and ADASYN-ENN, have shown significant improvements in handling imbalanced datasets in credit card fraud detection. The choice of technique employed should take into account the dataset properties, computational constraints, and the desired balance between precision and recall. Ensemble-based hybrid techniques like SMOTE-ENN provide the best performance but come at a higher computational cost.

Conclusion

The primary goal of this work is to improve credit card fraud detection performance by tackling the problems associated with biased predictions, overfitting, and imbalanced datasets. The increasing use of credit cards has led to a rise in credit card fraud; hence, it is critical to create efficient fraud detection algorithms. The study used an oversampling technique, more precisely, a hybridized strategy that combines the Synthetic Minority Over-sampling Technique (SMOTE) and Edited Nearest Neighbor (ENN) to balance the dataset to do this.

The goals of the study are to gather a dataset of credit card fraud transactions, balance the dataset using the SMOTE-ENN hybrid technique, create a deep neural network for the purpose of detecting credit card fraud, and then test and assess the model's effectiveness. The credit card fraud dataset's minority class was oversampled using the hybrid data balancing method known as SMOTE-ENN. Using SMOTE, synthetic data points were created for the minority class, and ENN was used to accurately classify every observation in the minority class. As a result, each class had an equal number of data points in the balanced dataset.

The results of this investigation highlight how critical it is to address the issues related to unbalanced datasets in credit card fraud detection. Financial transactions are at serious risk due to the possibility of biased forecasts resulting from an imbalanced dataset. The investigation showed that the prediction model's biased precision was caused by the initial unbalanced dataset. Overfitting and biased predictions were, however, successfully addressed by the addition of dropout layers to the deep neural network architecture and the use of SMOTE-ENN for data balancing. The model addressed the difficulties caused by unbalanced data in credit card fraud detection by achieving a balanced prediction with noticeably better performance.

References

[1]. Adebayo, O. S., Favour-Bethy, T. A., Otasowie, O., & Okunola, O. A. (2023). Comparative review of credit card fraud detection using machine learning and concept drift techniques. *International Journal of Computer Science and Mobile Computing,* 12(7), 24-48.

https://doi.org/10.47760/ijcsmc.2023.v12i07.004

[2]. Afriyie, J. K., Tawiah, K., Pels, W. A., Addai-Henne, S., Dwamena, H. A., Owiredu, E. O., & Eshun, J. (2023). A supervised machine learning algorithm for detecting and predicting fraud in credit card transactions. *Decision Analytics Journal,* 6, 100163.

https://doi.org/10.1016/j.dajour.2023.100163

[3]. Aftab, A. U., Shahzad, I., Anwar, M., Sajid, A., & Anwar, N. (2023). Fraud detection of credit cards using supervised machine learning. *Pakistan Journal of Emerging Science and Technologies (PJEST),* 4, 38-51.

https://doi.org/10.58619/pjest.v4i3.114

[4]. Alarfaj, F. K., Malik, I., Khan, H. U., Almusallam, N., Ramzan, M., & Ahmed, M. (2022). Credit card fraud detection using state-of-the-art machine learning and deep learning algorithms. *IEEE Access,* 10, 39700-39715.

https://doi.org/10.1109/ACCESS.2022.3166891

[5]. Alenzi, H. Z., & Aljehane, N. O. (2020). Fraud detection in credit cards using logistic regression. *International Journal of Advanced Computer Science and Applications,* 11(12), 1-13.

[6]. Alfaiz, N. S., & Fati, S. M. (2022). Enhanced credit card fraud detection model using machine learning. *Electronics,* 11(4), 662.

https://doi.org/10.3390/electronics11040662

[7]. Alharbi, A., Alshammari, M., Okon, O. D., Alabrah, A., Rauf, H. T., Alyami, H., & Meraj, T. (2022). A novel text2IMG mechanism of credit card fraud detection: A deep learning approach. *Electronics,* 11(5), 756.

https://doi.org/10.3390/electronics11050756

[8]. Abd El-Naby, A., Hemdan, E. E. D., & El-Sayed, A. (2023). An efficient fraud detection framework with credit card imbalanced data in financial services. *Multimedia Tools and Applications,* 82(3), 4139-4160.

https://doi.org/10.1007/s11042-022-13434-6

[9]. Esenogho, E., Mienye, I. D., Swart, T. G., Aruleba, K., & Obaido, G. (2022). A neural network ensemble with feature engineering for improved credit card fraud

detection. *IEEE Access,* 10, 16400-16407.

https://doi.org/10.1109/ACCESS.2022.3148298

[10]. Dhiman, D., Bisht, A., Kumari, A., Anandaram, H., Saxena, S., & Joshi, K. (2023). Online fraud detection using machine learning. In 2023 *International Conference on Artificial Intelligence and Smart Communication (AISC)* (pp. 161-164). IEEE.

https://doi.org/10.1109/AISC56616.2023.10085493

[11]. Gupta, P., Varshney, A., Khan, M. R., Ahmed, R., Shuaib, M., & Alam, S. (2023). Unbalanced credit card fraud detection data: A machine learning-oriented comparative study of balancing techniques. *Procedia Computer Science,* 218, 2575-2584.

https://doi.org/10.1016/j.procs.2023.01.231

[12]. Hordri, N. F., Yuhaniz, S. S., Azmi, N. F. M., & Shamsuddin, S. M. (2018). Handling class imbalance in credit card fraud using resampling methods. *International Journal of Advanced Computer Science and Applications,* 9(11), 390-396.

[13]. Medida, J., Bharath Reddy, Y. S., Priya, K. C., Vardhan Reddy, M. H., & Prashanth, K. S. (2024). A deep learning ensemble with data resampling for credit card fraud detection. *International Journal for Innovative Engineering & Management Research,* 13(4).

[14]. Khalid, A. R., Owoh, N., Uthmani, O., Ashawa, M., Osamor, J., & Adejoh, J. (2024). Enhancing credit card fraud detection: An ensemble machine learning approach. *Big Data and Cognitive Computing,* 8(1), 6.

https://doi.org/10.3390/bdcc8010006

[15]. Muntasir Nishat, M., Faisal, F., Jahan Ratul, I., Al-Monsur, A., Ar-Rafi, A. M., Nasrullah, S. M., & Khan, M. R. H. (2022). A comprehensive investigation of the performances of different machine learning classifiers with SMOTE-ENN oversampling technique and hyperparameter optimization for imbalanced heart failure dataset. *Scientific Programming,* 2022(1), 3649406.

https://doi.org/10.1155/2022/3649406

[16]. Sasank, J. S., Sahith, G. R., Abhinav, K., & Belwal, M. (2019). Credit card fraud detection using various classification and sampling techniques: A comparative study. In 2019 *International Conference on Communication and Electronics Systems (ICCES)* (pp. 1713-1718). IEEE.

https://doi.org/10.1109/ICCES45898.2019.9002289

[17]. Shamsudin, H., Yusof, U. K., Jayalakshmi, A., & Khalid, M. N. A. (2020). Combining oversampling and undersampling techniques for imbalanced classification: A comparative study using credit card fraudulent transaction dataset. In 2020 *IEEE 16th International Conference on Control & Automation (ICCA)* (pp. 803-808). IEEE.

https://doi.org/10.1109/ICCA51439.2020.9264517

[18]. Udeze, C. L., Eteng, I. E., & Ibor, A. E. (2022). Application of machine learning and resampling techniques to credit card fraud detection. *Journal of the Nigerian Society of Physical Sciences,* 769-769.

https://doi.org/10.46481/jnsps.2022.769

[19]. Sisodia, D. S., Reddy, N. K., & Bhandari, S. (2017). Performance evaluation of class balancing techniques for credit card fraud detection. In 2017 *IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)* (pp. 2747-2752). IEEE.

https://doi.org/10.1109/ICPCSI.2017.8392219

[20]. Mishra, A., & Ghorpade, C. (2018). Credit card fraud detection on the skewed data using various classification and ensemble techniques. In 2018 *IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)* (pp. 1-5). IEEE.

https://doi.org/10.1109/SCEECS.2018.8546939

[21]. Ahammad, J., Hossain, N., & Alam, M. S. (2020). Credit card fraud detection using data pre-processing on imbalanced data-Both oversampling and undersampling. In *Proceedings of the International Conference on Computing Advancements* (pp. 1-4).

https://doi.org/10.1145/3377049.3377113

[22]. Sadineni, P. K. (2020). Detection of fraudulent transactions in credit card using machine learning algorithms. In 2020 *Fourth International Conference on I-*

SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 659-660). IEEE.
https://doi.org/10.1109/I-SMAC49090.2020.9243545

[23]. Azhan, M., & Meraj, S. (2020). Credit card fraud detection using machine learning and deep learning techniques. In 2020 *3rd International Conference on Intelligent Sustainable Systems (ICISS)* (pp. 514-518). IEEE.
https://doi.org/10.1109/ICISS49785.2020.9316002

ABOUT THE AUTHORS

Enesi Femi Aminu is currently a Senior Lecturer in the Department of Computer Science, School of Information & Communication Technology, Federal University of Technology (FUT), Minna, Nigeria. He has taught numerous courses, including Artificial Intelligence, Expert Systems, Operating Systems, and Advanced Database Design. He obtained his Ph.D. in Computer Science from the Federal University of Technology, Minna, Nigeria. Additionally, he earned both his B.Sc. and M.Sc. degrees in Computer Science from the University of Jos, Jos, Nigeria, and Ahmadu Bello University, Zaria, Nigeria, respectively. His current research interests include Knowledge Representation, specifically Ontology Design and Semantic Search, Artificial Intelligence, and Machine Learning. He is a member of several professional bodies, including the Computer Professional Registration Council of Nigeria (CPN), Nigeria Computer Society (NCS), and International Association of Computer Science and Information Technology (IACSIT). He has published over 35 peer-reviewed scientific articles in reputable journals, international conferences, and book chapters.

Abdulqadri Olalekan Araoye was a student in the Department of Computer Science at the Federal University of Technology, Minna, Nigeria. This study forms part of his undergraduate research work.

Ayobami Ekundayo is a Lecturer in the Department of Computer Science at the Federal University of Technology, Minna, Nigeria. She is currently pursuing her Ph.D. in Computer Science at the Federal University of Technology, Minna. She holds an M.Sc. and a B.Sc. in Computer Science from the University of Ilorin, Ilorin, Kwara. Her specializations include Data Mining, Business Intelligence, Predictive Analytics, and Data Warehousing. She has published in well-regarded local and international journals.

Oluwaseun Adeniyi Ojerinde is a Senior Lecturer in the Department of Computer Science at the Federal University of Technology, Minna, Nigeria. He has over nine years of experience in research and undergraduate and graduate-level teaching. His research and publications encompass encryption and blockchain technology for data management, process optimization, machine learning, 5G technology, antenna systems and propagation, on-body systems, Multiple Input Multiple Output (MIMO) systems, Telecommunications, Specific Absorption Rate (SAR), Radiation, and Computer Science Education.

Grace Amina Onyeabor is a Senior Lecturer in the Department of Information Technology at the Federal University of Technology, Minna, Nigeria, where she supervises undergraduate and postgraduate students (Master's and Ph.D.). She holds a B.Tech in Physics/Computer Science from the Federal University of Technology, Minna; an M.Sc. in Information Science from the University of Ibadan, Nigeria; and a Ph.D. in Information Technology from Universiti Utara Malaysia. Her research interests include data quality, data engineering, data analysis, data management, big data, and data science. She is a member of the Nigeria Computer Professionals.