ABSTRACT

This study was aimed at the development of a predictive model for the detection of online fraud among online transactions. The study collected a dataset containing information about legitimate and fraudulent transactions from a fraud detection company. The study analysed the properties of the dataset and applied the use of machine learning algorithms for the development of a fraud detection system. The study's results revealed that among the algorithms adopted, the XG boost showed the best performance. The performance of the XG boost was further improved following hyperparameter tuning. The selected XG boost was deployed for use as an API required for the detection of online fraud.

Keywords: online fraud detection, machine learning, cloud-based frameworks

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY OF TERMS

AI	Artificial Intelligence
ANN	Artificial Neural Network
AWS	Amazon Web Service
CNN	Convolutional Neural Network
DevOps	Development and Operations
DNN	Deep Neural Network
DT	Decision Trees
FCNN	Fully Convolutional Neural Network
GBM	Gradient Boosting Machine
GCP	Google Cloud Platform
IaaS	Infrastructure-as-a-Service
KNN	k-Nearest Neighbour
LR	Logistic Regression
LSTM	Long Short-Term Memory
ML	Machine Learning
MLOps	Machine Learning Operations
NB	Naïve Bayes
PaaS	Platform-as-a-Service
PCA	Principal Component Analysis
RF	Random Forest
SVM	Support Vector Machine

1. Introduction

Financial fraud is obtaining financial gains by dishonest and illegal means (Hilal, *et al*., 2021; Ashtiani & Raahemi, 2021). Financial fraud can occur in various contexts, including the business, banks, insurance, and taxes sectors (Albashrawi, 2016). Tax evasion, money transfer fraud, and other financial crime have become progressively more of a burden for industries and corporations (Choi & Lee, 2018; Ngai, *et al*., 2011; Choi & Lee, 2018). Financial fraud, a dishonest strategy used to obtain monetary advantages, has emerged as a more pervasive threat to businesses. Despite several initiatives to curtail financial fraud, it continues to harm society and the economy, as significant sums of money are lost to fraud every day (Ryman-Tubb, *et al*., 2018).

Several methods for detecting fraud were first presented some years ago (Hilal, *et al*., 2021). Most old procedures are manual, time-consuming, expensive, inaccurate, and unworkable (Al-Hashedi & Magalingam, 2021). Traditional methods for identifying and preventing fraudulent activities employ inaccurate, expensive, and time-consuming human validation and examination processes. Additional research is being done; however, they are ineffective at reducing losses brought on by fraud (Ngai, et al., 2011). Machine learning and data mining are used to identify suspicious transactions in the finance industry thanks to the development of artificial intelligence (AI) (Da'U & Salim, 2019; Zeng & Tang, 2021).

2. Problem Description

This research focused on using a cloud-based data analytics and programming framework to implement AI applications required to detect online fraud based on mobile device customer transactions. This was done to demonstrate the capabilities of the adoption of AI and cloud environments for solving real-life problems. This study attempts to provide a model which a fraud detection company can use to detect online fraud based on information about customer transactions. The successful detection of fraudulent transactions by the developed model will attract monetary charges by the customers to the company as a service charge. There is a need to develop an effective and efficient predictive model that will aid in the detection of online financial fraud.

3. Project Motivation and Objective

This research was motivated by the recent adoption of cloud-based AI frameworks for developing predictive models using machine learning algorithms for solving various real-life problems, especially among financial transactions from which fraud can be detected. The study investigates AI and cloud services' role in fraud analysis, detection, and prevention. The research objectives are to select a relevant data source, explore the characteristics of the information stored in the dataset, and implement an ML pipeline using a suitable cloud-based AI framework.

4. Research Challenges

There is a need to develop a production model for the CEO of Blocker Fraud Company which can be accessed via an API that can be used to assess the transactions made by customers, thereby classifying them as either fraudulent or legitimate.

5. Literature Review

Mayekar, *et al.* (2021), worked on developing a fraud detection model for online transactions using machine learning. The study collected a dataset containing information about fraud and legitimate related features. The study adopted the use of the XG boosting algorithm for the development of the predictive model. The study adopted the use of the Python jupyter environment for the model development using a single simulation which involved the use of 70% for training and 30% for testing. The results revealed that the XG boost achieved a performance of 99.83%. The study was limited to using a standalone system to develop the model and did not consider using a cloud-based AI framework for development.

Gonanoina and Muttipati (2021), worked on the application of machine learning algorithms for the detection of credit card fraud. The study collected a dataset containing information about credit card fraud, following which preprocessing techniques were applied to the dataset using principal component analysis (PCA). The study applied logistic regression, random forest and categorical boosting algorithms to develop the predictive model for fraud detection. The study's results revealed an accuracy of 99.95% using the random forest algorithm. However, the study did not consider using a cloud-based AI framework.

Gottumukkala and Rao (2020) worked on developing a fraud detection model using a cloud-based AI framework. The study collected a dataset from Kaggle containing fraudulent and legitimate data records. The study adopted Fully Convolutional Neural Networks (FCNN) and gradient-boosting machine learning algorithms. The model was compared with existing machine learning algorithms: support vector machines (SVM), logistic regression, and artificial neural networks (ANN). The study's results revealed that the proposed FCNN-GBM algorithms performed better than existing supervised machine learnings.

Suvarna and Kowshalya (2020), worked on applying a federation of machine learning algorithms to develop a predictive model for the detection of credit card fraud. The study collected the dataset from a repository following a cloud-based server that contained the dataset connected to a number of workers. The dataset was moved to the worker servers to build the predictive model using the auto-encoder and radial basis function (RBF). The results of the model developed by the 3 workers were later fed to the federated server, which was used to create the final model. The study's results revealed that federated learning was only able to improve the level of security but failed to improve the machine learning algorithm's performance.

Garcia *et al.* (2016), worked on developing a cloud-based framework for facilitating machine learning simulations. The study reviewed related works on machine learning, following which the motivation for machine learning as a service platform was provided. The study adopted the DEEP framework, which allowed machine learning algorithms to be deployed on e-Infrastructures. The cloud-based framework comprised the DEEP open catalogue, DEEP learning facility, DEEP as a service, and storage and data services. The implemented framework was adopted for use in the classification of plant micros-organisms. The results of the development of the model revealed good performance.

5. Cloud-Based AI Framework for MLOps

Major corporations use machine learning operations (MLOps), and best practices to effectively run AI solutions with the aid of an ever-increasing selection of software and cloud services. AI aids in various tasks, including maintaining stock inventory, analysing x-ray images, estimating credit risks, and translating web pages (Merritt, 2020). Since machine learning's processing is comparable to tasks carried out by IT systems, it has now achieved the same level of mainstream acceptance as software development. MLOps is based on the

profession of DevOps, which is currently in use and refers to the effective creation, deployment, and operation of business apps. (Breuel, 2022). DevOps began as a mechanism for rival software development teams (the Devs) and IT operations teams (the Ops) to work together ten years ago. Data scientists who create datasets and AI models to analyse them are added to the team by MLOps. It also comprises ML engineers that employ structured, automated processes to pass the datasets through the models (see Figure 1).

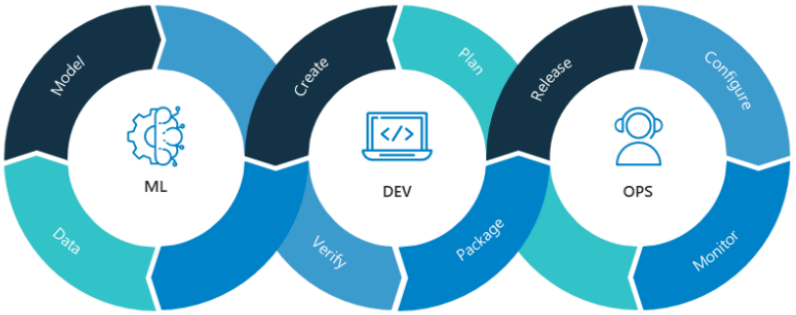

Figure 1. The combination of MLOps with application development and IT operations
(Source: Merritt, 2020)

An enterprise data centre can add the following components of an MLOps software layer once it has an AI infrastructure set up data sources and the datasets produced from them; an archive of AI models labelled with their histories and characteristics; an automated ML pipeline that maintains datasets, models, and studies through their life - cycle; and application containers, based mainly on Kubernetes, to make running such jobs easier. (Merritt, 2020). Data scientists require the freedom to use adaptable sandboxes and repositories to track and update datasets across providers continuously. Figure 2 depicts an example of an ML pipeline. They collaborate with ML engineers who create prototype models from business datasets and test and refine them. These features are now included in cloud computing services, and many businesses are employing these services to build their centres of excellence for artificial intelligence.

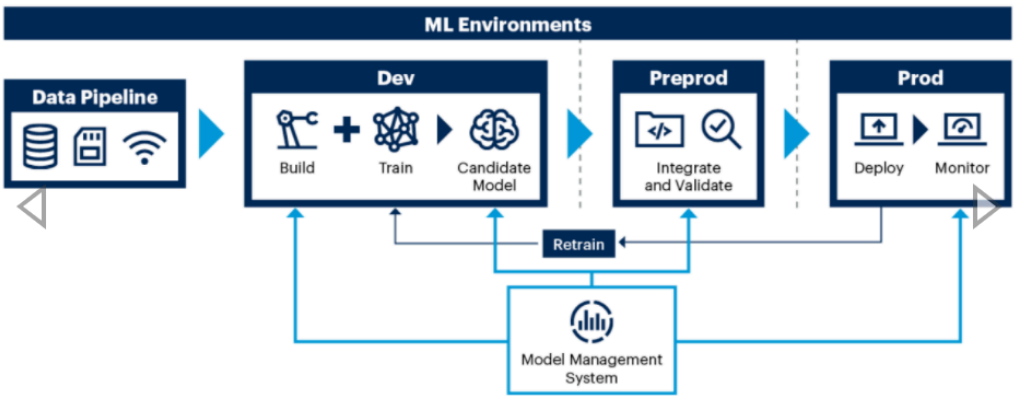

Figure 2. A typical AI pipeline
(Source: Gartner, 2019)

5.1 ML Pipelines

Since it involves several modifications conducted between the source of data and its target, a data pipeline is a crucial component of data engineering. They are typically depicted as graphs, where each node represents a modification, and each edge represents a dependency or sequence of execution (Breuel, 2022). ETL (extract, transform, and load) pipelines are another name for data pipelines. By developing two pipelines, one for constructing the model and the other for using it, the steps in an ML process may be incorporated into the data pipeline, turning it into an ML pipeline. The ML pipeline connects code and data planes in an organised and systematic way since it is a pure code construct separate from the information instances (see Figure 3 for the ML pipeline).

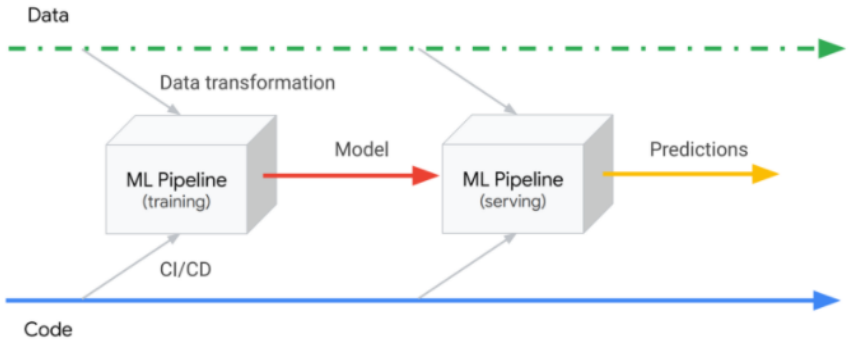

Figure 3. ML pipeline for connecting data and code for producing models and predictions
(Source: Breuel, 2022)

5.2 Google Cloud Platform (GCP)

GCP, a suite of cloud computing services provided by Google, utilises the same internal architecture as Google's end-user products (Weinberger, 2022). Additionally, it includes a collection of management tools that may be used to deliver services, including computing, data storage, data analytics, and machine learning. GCP provides

infrastructure-as-a-service (IaaS), Platform-as-a-Service (PaaS) and serverless computing environments (Google, 2022). The various products made available by Google under Cloud AI include BigQuery, Cloud Dataflow, Cloud Data Fusion, Dataproc, Cloud Datalab, which serves Google Colab, Cloud Data Studio etc.

6. Data Description

This study collected data containing information about transactions that customers have performed. The data was collected from an online repository provided by Kaggle from the URL https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset?resource=download. The information was provided by Blocker Fraud Company which specialises in the detection of fraud in financial transactions that were conducted via mobile devices. The dataset is composed of 11 feature attributes containing information about 6,362,620 records. Following is a description of the feature attributes contained within the collected dataset: Among the identified feature attributes in the dataset, it was observed that some variables were numeric while some are nominal. The nominal variables in the dataset are type, nameOrig, and nameDest. The numeric variables consist of integer-valued variables, namely: step, isFraud and isFlaggedFraud, alongside floating-point valued variables, namely: amount, oldBalanceOrig, newBalanceOrig, oldBalanceDest and newBalanceDest.

7. Data Preparation

The analysis required in this study was conducted using a Mac OS with 16GB RAM and 1TB ROM using jupyter notebook alongside several packages and libraries that were adopted for the execution of the tasks. The joblib is a function that contains a set of tools used to provide lightweight pipelining in Python and was integrated into the environment using the command *import joblib*. It provided a better way to avoid recomputing the same function repetitively, saving a lot of time and computational costs. Figure 4 shows the source codes for the various functions imported into the notebook.

```
import joblib
import warnings
import inflection
import time

import numpy as np
import pandas as pd
import seaborn as sns

import matplotlib.pyplot as plt

from scipy import stats
from boruta import BorutaPy
from category_encoders import OneHotEncoder

from IPython.display import Image
from IPython.core.display import HTML

from xgboost import XGBClassifier
from lightgbm import LGBMClassifier

from sklearn.svm import SVC
from sklearn.dummy import DummyClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import balanced_accuracy_score, precision_score, classification_report
from sklearn.metrics import recall_score, f1_score, make_scorer, cohen_kappa_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import GridSearchCV, train_test_split, StratifiedKFold


#from churn.Churn import Churn
from flask import Flask, request, Response


import requests
```

Figure 4. Python functions imported for data analysis

Other custom helper functions were created apart from the built-in Python function to be called by the application via the API. A function *jupyter_settings* was created to display the settings of the jupyter notebooks by the system user. Figure 5 shows the source code of the function *jupyter_settings.*

```
def jupyter_settings():
    %matplotlib inline
    %pylab inline

    sns.set(font_scale=1.6)

    plt.style.use('seaborn-darkgrid')
    plt.rcParams['figure.figsize'] = [15, 7]
    #plt.rcParams['figure.figsize'] = [25, 12]
    plt.rcParams['font.size'] = 16

    display(HTML('<style>.container { width:100% !important; }</style>'))
    pd.options.display.max_columns = None
    pd.options.display.max_rows = None
    pd.set_option('display.expand_frame_repr', False)

jupyter_settings()
```

Figure 5. Source code for function jupyter_settings

Function *ml_scores* which took as input the name of the model alongside the actual and predicted outputs was created for evaluating the performance of the model generated by a selected machine learning algorithm. Figure 6 shows the source code of the function *ml_scores.*

```
def ml_scores(model_name, y_true, y_pred):

    accuracy = balanced_accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred)
    recall = recall_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred)
    kappa = cohen_kappa_score(y_true, y_pred)

    return pd.DataFrame({'Balanced Accuracy': np.round(accuracy, 3),
                         'Precision': np.round(precision, 3),
                         'Recall': np.round(recall, 3),
                         'F1': np.round(f1, 3),
                         'Kappa': np.round(kappa, 3)},
                        index=[model_name])
```

Figure 6. Source code for the function ml_scores

Function *calcCramerV,* which takes two values as input, was created to assess the degree of association between two categorical fields based on the chi-square test of independence. These two categorical values were required to capture data containing input features and target variables. Figure 7 shows the source code of the function *calcCramerV.* The function *ml_cv_results* takes the model's name, model, input, and output dataset as inputs. It was required for performing various functions, including data normalisation, cross-validation, splitting the data into the training and testing dataset, and showing the performance evaluation results. This function served as the primary function to call every other user-defined function. Figure 8 shows the source code of the function *ml_cv_results*.

```
def calcCramerV(x, y):
    cm = pd.crosstab(x, y).values
    n = cm.sum()
    r, k = cm.shape

    chi2 = stats.chi2_contingency(cm)[0]
    chi2corr = max(0, chi2 - (k-1)*(r-1)/(n-1))

    kcorr = k - (k-1)**2/(n-1)
    rcorr = r - (r-1)**2/(n-1)

    return np.sqrt((chi2corr/n) / (min(kcorr-1, rcorr-1)))
```

Figure 7. Source code for the function calcCramerV

```python
def ml_cv_results(model_name, model, x, y, verbose=1):

    '''initial'''
    balanced_accuracies = []
    precisions = []
    recalls = []
    f1s = []
    kappas = []

    mm = MinMaxScaler()

    x_ = x.to_numpy()
    y_ = y.to_numpy()

    count = 0

    '''cross-validation'''
    skf = StratifiedKFold(n_splits=5, shuffle=True)

    for index_train, index_test in skf.split(x_, y_):
        ## Showing the Fold
        if verbose > 0:
            count += 1
            print('Fold K=%i' % (count))

        ## selecting train and test
        x_train, x_test = x.iloc[index_train], x.iloc[index_test]
        y_train, y_test = y.iloc[index_train], y.iloc[index_test]

        ## applying the scale
        x_train = mm.fit_transform(x_train)
        x_test = mm.transform(x_test)

        ## training the model
        model.fit(x_train, y_train)
        y_pred = model.predict(x_test)

        ## saving the metrics
        balanced_accuracies.append(balanced_accuracy_score(y_test, y_pred))
        precisions.append(precision_score(y_test, y_pred))
        recalls.append(recall_score(y_test, y_pred))
        f1s.append(f1_score(y_test, y_pred))
        kappas.append(cohen_kappa_score(y_test, y_pred))


    '''results'''
    accuracy_mean, accuracy_std = np.round(np.mean(balanced_accuracies), 3), np.round(np.std(balanced_accuracies), 3)
    precision_mean, precision_std = np.round(np.mean(precisions), 3), np.round(np.std(precisions), 3)
    recall_mean, recall_std = np.round(np.mean(recalls), 3), np.round(np.std(recalls), 3)
    f1_mean, f1_std = np.round(np.mean(f1s), 3), np.round(np.std(f1s), 3)
    kappa_mean, kappa_std = np.round(np.mean(kappas), 3), np.round(np.std(kappas), 3)

    ## saving the results in a dataframe
    return pd.DataFrame({"Balanced Accuracy": "{} +/- {}".format(accuracy_mean, accuracy_std),
                         "Precision": "{} +/- {}".format(precision_mean, precision_std),
                         "Recall": "{} +/- {}".format(recall_mean, recall_std),
                         "F1": "{} +/- {}".format(f1_mean, f1_std),
                         "Kappa": "{} +/- {}".format(kappa_mean, kappa_std)},
```

Figure 8. Source code for the function ml_cv_results

Following these tasks, the dataset required for this study was imported into the environment; however, due to the large nature of the dataset, only 10% of the total actual records were imported, leading to 636,262 records. Figure 9 shows the source code for data importation and a view of the dataset's first and last set of records. The dataset was imported as a data frame called *df1*.

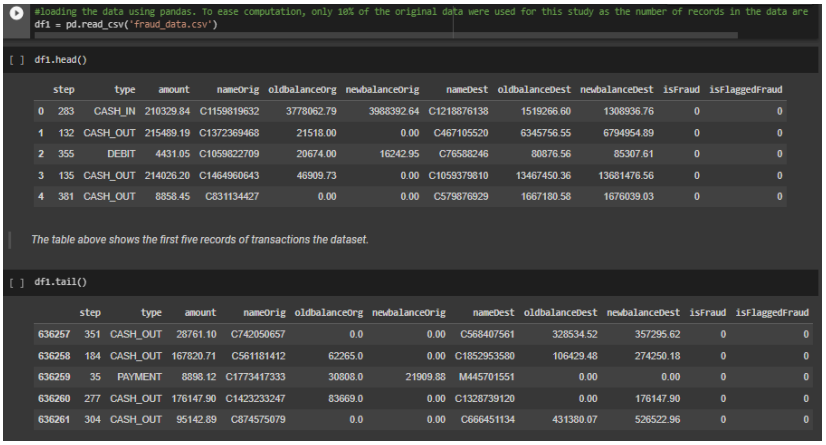

Figure 9. Source code for the function ml_cv_results

8. Data Preprocessing

The first major task was to rename the feature attributes in the dataset, especially those that appeared in camel case form, into underscored features using the inflection function. The assessment of the dataset revealed that there were no missing records in the dataset. The variables called *is_fraud* and *is_flagged_fraud,* which were integer-type variables, were converted into categorical values such that the values 1 and 0 were used to represent the values yes and no, respectively. Furthermore, the variables were classified into two main categories with the names cat_attributes and num_attributes for collecting the categorical and numeric attributes, respectively. Figure 10 shows the source code and results of the conversion of the names of the attributes alongside their groupings.

```
[ ] #Renaming the columns in the dataset
    cols_old = df1.columns.tolist()

    snakecase = lambda x: inflection.underscore(x)
    cols_new = list(map(snakecase, cols_old))

    df1.columns = cols_new

[ ] df1.columns

    Index(['step', 'type', 'amount', 'name_orig', 'oldbalance_org',
           'newbalance_orig', 'name_dest', 'oldbalance_dest', 'newbalance_dest',
           'is_fraud', 'is_flagged_fraud'],
          dtype='object')
```

Figure 10. Source code for the process of renaming columns

9. Feature Engineering

Some feature engineering tasks were performed on the collected dataset; however, a copy of the data frame was generated from the original data frame called *df2*. Afterwards,

an attribute called step was used to create the *step_days* and *step_weeks,* which were used to convert the hour value into a day of the week and week in a year, respectively. The difference between the initial balance before the transaction and the final balance after the transaction for the originator was determined, and it was used to generate the *diff_new_old_balance*. The difference between the initial balance before the transaction and the final balance after the transaction for the recipient was determined, and it was used to generate the *diff_new_old_dest*. Figure 11 shows the source code alongside the result of the feature engineering process.

```
[ ] df1['is_fraud'] = df1['is_fraud'].map({1: 'yes', 0: 'no'})
    df1['is_flagged_fraud'] = df1['is_flagged_fraud'].map({1: 'yes', 0: 'no'})

[ ] num_attributes = df1.select_dtypes(exclude='object') #selecting the numerical variables
    cat_attributes = df1.select_dtypes(include='object') #selecting the categorical variables
```

Figure 11.Source code for the data conversion process

10. Algorithms

Some supervised machine learning algorithms were adopted in this study to develop the predictive model required for detecting online fraud; they are stated as follows. K-Nearest Neighbours (KNN) is a supervised machine learning algorithm that adopts a distance (similarity) procedure by using a neighbouring class of similar records as a basis of the classification (Cover & Hart, 1967). Hence, the smaller the distance between two points, the more likely they are similar (Jaskowiak & Campello, 2011). The use of a distance metric determined the neighbours. Equation (1) shows the expression for the Euclidean distances adopted in this study. Assuming a dataset containing information about a pair (x_i, y_i) for *n* records, then the following holds:

$$Euclidean\ distace\ =\ d(x,y)\ =\sqrt{\sum_{i=1}^{n}\left(x_i - y_i\right)^2} \tag{1}$$

Support Vector Machines (SVM) is a supervised machine learning algorithm applied for the classification task (Bennett & Campbell, 2000). It maps them into the same area to determine what class new examples belong to (Polson & Scott, 2011). A space known as a dimensional hyperplane was used to produce the separation. The hyperplane with the most significant distance from any group's closest training data point achieves a decent separation. This suggests that the generalisation error of the SVM classifier will decrease with an increasing margin (Fradkin & Muchnik, 2006). Logistic regression is a machine

learning algorithm that calculates the likelihood that a binary event will occur based on a predetermined input data set. The dependent data attribute is anticipated based on an investigation of the link within some or all of the existing variables. This is the rationale behind selecting this model to choose features and determine whether a transaction is legitimate or fraudulent. A light gradient boosting machine (GBM) constructs an additive model of simple decision trees that are usually not well-optimised. These are generalised by optimising a pre-defined loss function, thus making better predictions (McCarthy, Kim, & Lee, 2020). The training dataset was generated as a proportion of the original dataset to build the predictive model from a dataset using the LightGBM model.

Cross-validation is a model validation technique for assessing how well statistical analysis results can generalise to an independent dataset (Piryonesi & El-Diraby, 2020).. In the k-fold cross-validation method, a dataset is divided into k equal parts (partitions), of which k-1 parts have been used to train the model, and the final part is used to assess the model's performance (Cawley & Talbot, 2010). See Figure 12 for an illustration of how the k-fold cross-validation methods work. In this study, 5-fold cross-validation cycles were carried out using various sections to lessen unpredictability. The models developed using the selected machine learning algorithms were compared based on performance evaluation metrics, such as accuracy, precision, recall, and f1 measure.

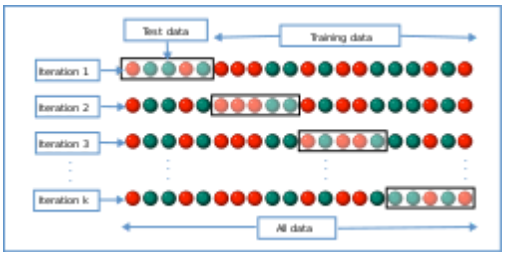

Figure 12. An illustration of the k-fold cross-validation approach

11. Results of Data Analysis and Visualization

The first part of the dataset analysis involved the univariate analysis of the distribution of the data records for the various attributes. The variable called *is_Fraud* was adopted as the dataset's response (or target) variable. The distribution of the records in the dataset revealed that 99.9% of the transactions were legitimate, while 0.1% were fraudulent transactions. Figure 13 shows the graphical plot of the distribution of the records for the variable *is_Fraud*.

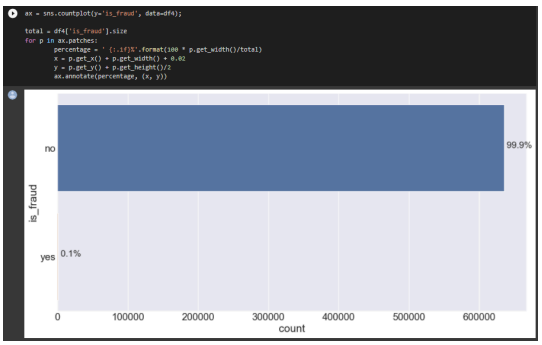

Figure 13. Distribution of the records in is_Fraud

The distribution of the records stored for the numerical attributes revealed that they were majorly right skewed as most of the dataset lay towards the left-hand side of the median value. Figure 14 shows the distribution of the numerical attributes in the dataset showing the skewness of the records stored in each attribute.

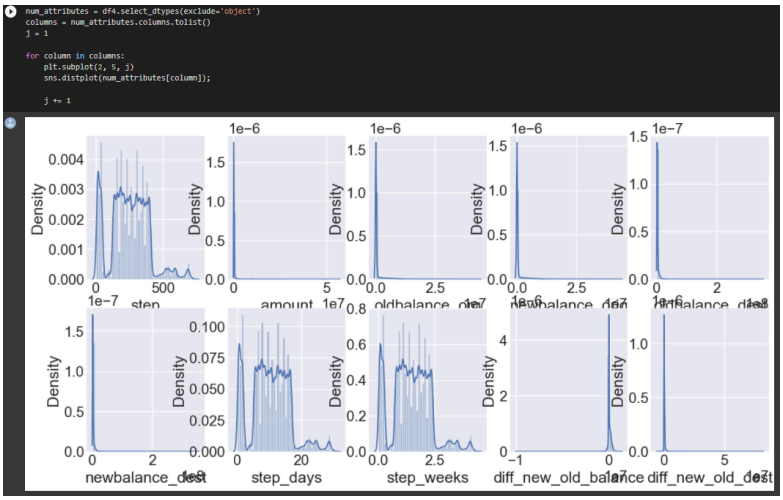

Figure 14. Distribution of the numerical input attributes

The distribution of the type of transactions carried out among the fraudulent transactions revealed that most of the frauds occurred via transfer, with a proportion of 50.4%. In comparison, the remaining 49.6% were carried out as cash_out transactions. Figure 15 shows the graphical plot of the distribution of the type of transactions performed across the fraudulent transactions. The distribution of the various types of transactions revealed that the majority of the transactions were cash out, with a proportion of 35.2%, followed by payments, with a proportion of 33.8%, cash in, with a proportion of 21.9%; transfers, with a proportion of 8.3% and debit with a proportion of 0.7%. Figure 16 shows the distribution of the types of transactions as presented in the collected dataset.

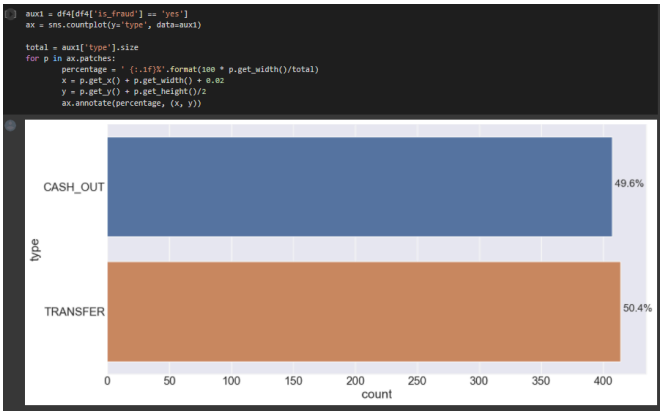

Figure 15. Distribution of the type of transactions among fraudulent transactions

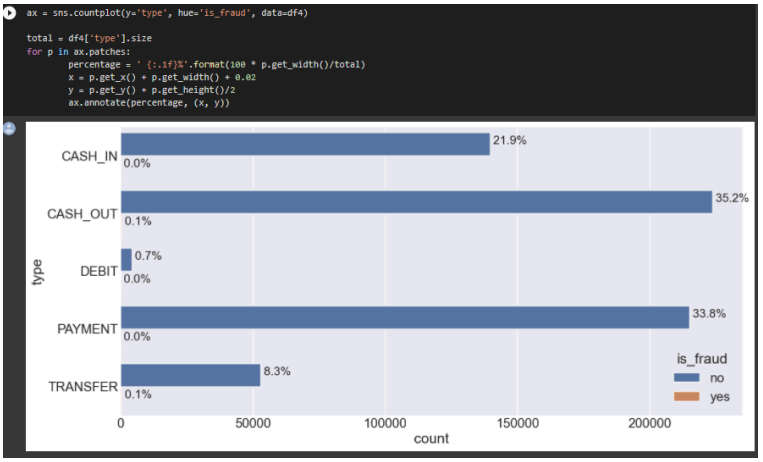

Figure 16. Distribution of the various types of transactions in the dataset

The second part of the analysis involved the multivariate analysis of the numerical and categorical attributes in the dataset. The analysis of the degree of association between the numerical attributes revealed a number of outcomes. Figure 17 shows the source code and results of the correlation analysis between the numerical features.

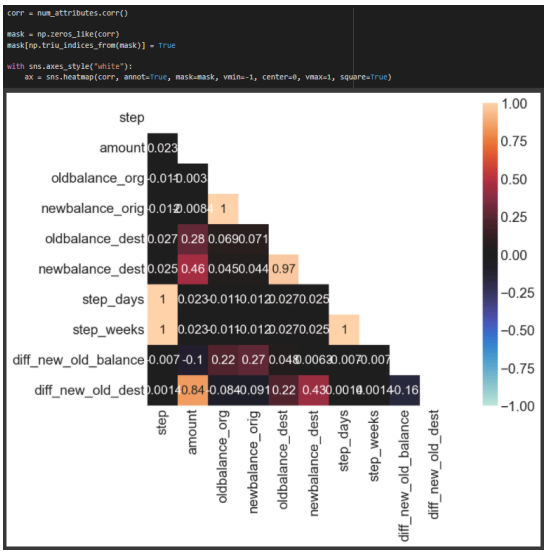

Figure 17. Analysis of the association between numerical attributes

The results revealed that a high degree of association exists between the new and old balance at the destination (0.97). A high degree of association exists between the amount and the difference between new and old transactions at destinations (0.84). In contrast, a moderate degree of association exists between the amount and new balance at the destination (0.46), a moderate degree of association between the new balance at the destination and the difference between the new and old balance at the destination (0.43). At the same time, the other numerical attributes have a very low degree of association.

The last part of the analysis involved the use of supervised machine learning algorithms to build the predictive models that were required for the detection of online fraud. The variables adopted as the input variables for the online fraud detection were determined by removing the variables is_fraud, is_flagged_fraud, name_orig, name_dest, step-weeks and step_days. In contrast, the target variable was identified as the variable called is_fraud. The dataset was split into three groups, namely: the training, validation and testing datasets using a proportion of 64%, 16% and 20%, respectively and by applying a 5-fold cross-validation technique. Figure 18 shows the source code and results of splitting the dataset.

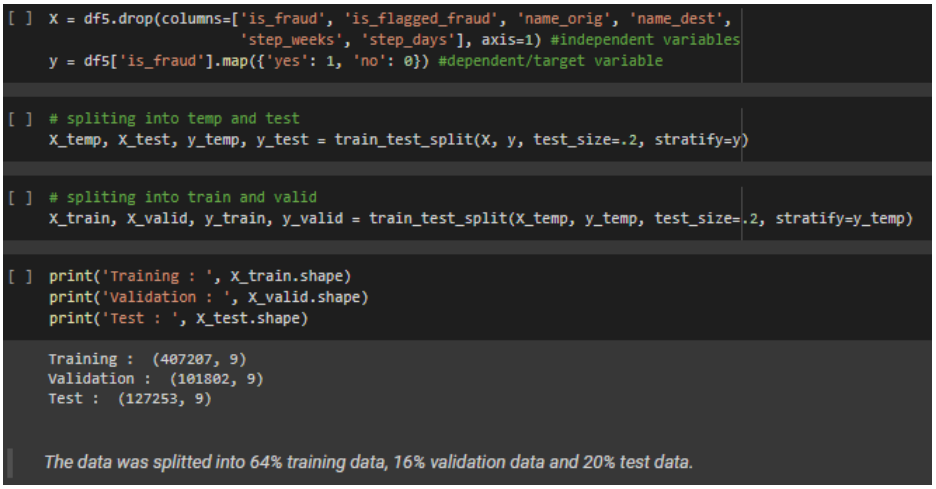

Figure 18. Source code and results of splitting the dataset

The machine learning algorithms that were adopted included a dummy, support vector machines (SVM), light gradient boosting machines (GBM), K-nearest neighbours (KNN), logistic regression (LR), random forest (RF) and XGBoost algorithms. The algorithms were adopted for generating the predictive model using the percentage split, which involved a single simulation, and the 5-fold cross-validation technique, which involved using 5 simulations. For both simulation processes, the average values of the precision, recall and f1

score were determined from the values estimated for the legitimate and fraudulent transactions. Table 1 shows the results of the single simulation adopted for the dataset using the selected supervised machine learning algorithms considered in this study.

Table 1. Results of single simulation using machine learning algorithms

Algorithm	Balanced Accuracy	Precision	Recall	F1
Dummy	0.500	0.000	0.000	0.000
SVM	0.500	0.000	0.000	0.000
Light GBM	0.652	0.047	0.313	0.081
KNN	0.565	1.000	0.130	0.230
LR	0.584	1.000	0.168	0.288
RF	0.844	0.978	0.687	0.807
XG Boost	0.866	0.970	0.733	0.835

The results revealed that among the identified algorithms, the XG boost algorithms showed the best performance with the highest value of the balances accuracy, recall and f1 measure. In contrast, the random forest algorithm proved to have the best overall precision but did not do as well as the XG boost algorithm. Table 2 shows the results of using the 5-fold cross-validation technique for the dataset using the supervised machine learning algorithms selected in this study.

Table 2. Results of 5-fold cross-validation using machine learning algorithms

Algorithm	Balanced Accuracy	Precision	Recall	F1
Dummy	0.500 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
LR	0.565 ± 0.013	1.000 ± 0.000	0.129 ± 0.026	0.228 ± 0.041
Light GBM	0.688 ± 0.125	0.263 ± 0.195	0.388 ± 0.237	0.304 ± 0.223
SVM	0.596 ± 0.017	1.000 ± 0.000	0.192 ± 0.034	0.320 ± 0.047
KNN	0.705 ± 0.017	0.943 ± 0.033	0.411 ± 0.034	0.572 ± 0.038
RF	0.861 ± 0.019	0.969 ± 0.018	0.721 ± 0.038	0.827 ± 0.029
XG Boost	0.879 ± 0.026	0.953 ± 0.020	0.758 ± 0.051	0.843 ± 0.032

The simulation revealed each performance evaluation metric's outcome based on each fold used for the simulation procedure. As a result, the results yielded varying values, which were summarised based on each simulation's mean and standard deviation. The

results revealed the XGBoost showed the best overall performance in the 5-fold cross-validation simulation procedure. Similarly, it had the best mean values for the accuracy, recall and f1-measure, followed by the random forest and KNN algorithms. The XGBoost algorithm was selected among the machine learning algorithms, and it was used to access the validation dataset following a bit of adjustment to tune its hyperparameters. Figure 19 shows the source codes and results of the procedure of tuning the hyperparameters of the XG boost algorithm.

```
[ ]  f1 = make_scorer(f1_score)

     params = {
         'booster': ['gbtree', 'gblinear', 'dart'],
         'eta': [0.3, 0.1, 0.01],
         'scale_pos_weight': [1, 774, 508, 99]
     }

[ ]  tZero = time.time() #time at the beginning

     gs = GridSearchCV(XGBClassifier(),
                       param_grid=params,
                       scoring=f1,
                       cv=StratifiedKFold(n_splits=5))

     gs.fit(X_params_cs, y_temp)

     t = time.time()-tZero #difference of time between tZero and time at the end of the code
     print('Time to compute:', round(t,3), ' sec')

     Time to compute: 4178.101  sec

[ ]  best_params = gs.best_params_
     best_params

     {'booster': 'gbtree', 'eta': 0.3, 'scale_pos_weight': 1}

[ ]  best_params = {'booster': 'gbtree', 'eta': 0.3, 'scale_pos_weight': 1}

[ ]  gs.best_score_

     0.8558293989756468
```

Figure 19. Source codes and results of the hyperparameter tuning

The results of applying the adjusted XGBoost algorithm following hyperparameter tuning revealed that the algorithm's performance based on the performance evaluation metrics was improved. It was observed that the algorithm's performance for the single simulation did not show any improvements; however, upon adopting the 5-fold cross-validation, there was an improvement in its performance. Figure 20 shows the source code and results of the performance of the adjusted XG boost algorithm. Afterwards, the XG boosting algorithm was applied to an unseen validation dataset which showed that the

accuracy improved to a value of 90.5%, with precision, recall and f1-score of 0.943, 0.811 and 0.872, respectively.

```
[ ] xgb_gs = XGBClassifier(
        booster=best_params['booster'],
        eta=best_params['eta'],
        scale_pos_weight=best_params['scale_pos_weight']
    )

[ ] xgb_gs.fit(X_train_cs, y_train)

XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False, eta=0.3,
              eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
              grow_policy='depthwise', importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_bin=256, max_cat_threshold=64, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', ...)

[ ] y_pred = xgb_gs.predict(X_valid_cs)
```

8.1.2 Single Results

```
[ ] xgb_gs_results = ml_scores('XGBoost GS', y_valid, y_pred)
    xgb_gs_results
```

	Balanced Accuracy	Precision	Recall	F1	Kappa
XGBoost GS	0.866	0.97	0.733	0.835	0.835

8.1.3 Cross Validation

```
[ ] xgb_gs_cv = ml_cv_results('XGBoost GS', xgb_gs, X_temp_cs, y_temp)
    xgb_gs_cv

Fold K=1
Fold K=2
Fold K=3
Fold K=4
Fold K=5
```

	Balanced Accuracy	Precision	Recall	F1	Kappa
XGBoost GS	0.879 +/- 0.009	0.967 +/- 0.014	0.758 +/- 0.018	0.85 +/- 0.012	0.85 +/- 0.012

Figure 20. Source code and results of the performance of adjusted XG boost

12. Discussion of Results

The results revealed that the dataset contained more information about fraudulent transactions than about legitimate transactions. The results showed that using the 5-fold cross-validation technique to develop the predictive model yielded better performance than using a single simulation technique. The results of the predictive model development revealed that among the algorithms adopted, the XG boost algorithms proved to have the best overall performance. Furthermore, it was shown that adjusting the hyperparameters of the XG boost further improved the performance of the detection of online transactions. The

selection model was adopted to answer some questions by the Blocker Fraud company regarding monetising customer services.

The results of the assessment of the 25% received by the company based on the transaction correctly detected by fraud revealed a value of $60,494,597.65 to be collected, the results of the assessment of the 5% received by the company on transactions misclassified as fraud whereas legitimate revealed a value of $154,803.92 to be collected while the results of the assessment of the 100% returned by the company to the customer for transactions which were correctly predicted as fraudulent revealed a value of $4,022,816.35 to be paid. Figure 21 shows the source code and results of the analysis.

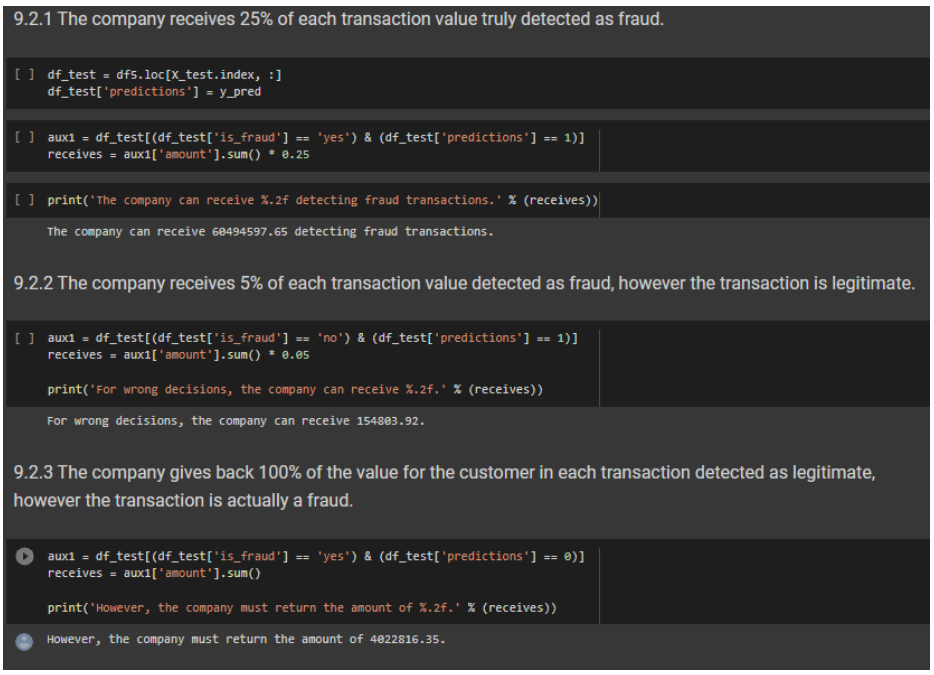

Figure 21. Source code and results of the payment received and paid for transactions

13. Conclusion

The study concluded that among the machine learning algorithms that were considered in this study that the XGBoost algorithm proved to have the best overall performance. The results of the feature selection algorithm revealed that the most important features were step, old balance from originator, new balance from originator, new balance from destination, difference in new and old balance from originator, difference in new and old balance on destination and type of transaction.

14. Legal and Ethical Considerations

This study does not violate any ethical regulation regarding data privacy and confidentiality, as the information provided regarding customer transactions used in the study had been anonymised already. The dataset adopted for use in this study was collected from a publicly available data repository and, as such, has not violated copyright or data ownership laws or protection against the unauthorised access and use of the dataset. The dataset has been made publicly available for use in research-based studies.

REFERENCES

Albashrawi, M., 2016. Detecting Financial Fraud Using Data Mining Techniques: A Decade Review from 2004 to 2015.. *Journal of Data Science,* Volume 14, pp. 553-570.

Al-Hashedi, K. & Magalingam, P., 2021. Financial fraud detection applying data mining techniques: A comprehensive review from 2009 to 2019. *Computational Science Review,* Volume 40, pp. 1-13.

Ashtiani, M. & Raahemi, B., 2021. Intelligent Fraud Detection in Financial Statements Using Machine Learning and Data Mining: A Systematic Literature Review.. *IEEE Access,* Volume 10, pp. 72504-72525.

Bennett, K. & Campbell, C., 2000. Support Vector Machine: Hype or Halleluyah. *SIGKDD Explorations,* 2(2), pp. 1-13.

Breuel, C., 2022. *MLOps: Machine Learning as an Engineering Discipline.* [Online]
Available at: https://builtin.com/machine-learning/mlops
[Accessed 2 January 2023].

Cawley, G. & Talbot, N., 2010. On Over-Fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *Journal of Machine Learning Research,* Volume 11, pp. 2079-2107.

Choi, D. & Lee, K., 2018. An Artificial Intelligence Approach to Financial Fraud Detection under IoT Environment: A Survey and Implementation. *Security Communication Network,* Volume 2018, pp. 1-15.

Cover, T. & Hart, P., 1967. Nearest Neighbour Pattern Classification. *IEEE Transactions on Information Theory,* 13(1), pp. 21-27.

Da'U, A. & Salim, N., 2019. Recommendation system based on deep learning methods: A systematic review and new directions.. *Artificial Intelligence Review,* Volume 53, pp. 2709-2748.

Devjver, P. & Kittler, J., 1982. *Pattern Recognition: A Statistical Approach.* London: Prentice-Hall.

Elfron, B. & Tibshirani, R., 1997. Improvements on Cross-Validation: The .632 + Boostrap Method. *Journal of the American Statistical Association,* 92(438), pp. 548-560.

Fradkin, D. & Muchnik, I., 2006. Support Vector Machines for Classification. In: *Discrete Methods in Epidemiology. DIMACS Series in Discrete Mathematics and Theoretical Computer Science.* s.l.:s.n., pp. 13-20.

Gonanoina, H. & Muttipati, A., 2021. Machine Learning Methods for discovering Credit Card Fraud. *International Research Journal of Computer Science (IRJCS),* 1(8), pp. 1-6.

Gottumukkala, P. & Rao, G., 2020. An Implementation of Real and Accurate Cloud-Based Fraud Detection System using Deep and Machine learning Algorithms. *International Journal of Advanced Research in Engineering and Technology,* 11(10), pp. 51-66.

Hilal, W., Gadsden, S. & Yawney, J., 2021. Financial Fraud: A Review of Anomaly Detection Techniques and Recent Advances.. *Expert System Applications,* p. 193.

Jaskowiak, P. & Campello, R., 2011. Comparing Correlation Coefficients as Dissimilarity Measures for Cancer Classification in Gene Expression Data. *Brazilian Symposium on Bioinformatics,* Volume 136, pp. 1-8.

Lee, Y., Lin, Y. & Wahba, G., 2001. Multicategory Support Vector Machines. *Journal of the American Statistical Association,* Volume 33, pp. 1-13.

Mayekar, V., Mattha, S., Choudhary, S. & Sankhe, A., 2021. Online Fraud Transaction Detection using Machine Learning. *International Journal of Engineering and Technology,* 8(5), pp. 645-649.

McCarthy, D., Kim, H. & Lee, H., 2020. Evaluation of Light Gradient Boosted Machine Learning Technique in Large Scale Land Use and Land Cover Classification. *Journal of Environments,* 7(84), pp. 1-20.

Merritt, R., 2020. *What is MLOps.* [Online]
Available at: https://blogs.nvidia.com/blog/2020/09/03/what-is-mlops/
[Accessed 2 January 2023].

Ngai, E. et al., 2011. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature.. *Decision Support Systems,* Volume 50, pp. 559-569.

Piryonesi, S. & El-Diraby, T., 2020. Data Analytics in Asset Management: Cost Effective Prediction of the Pavement Condition Index.. *Journal of Infrastructure Systems,* 26(1), pp. 23-34.

Platt, J., Christianini, N. & Shawe-Taylor, J., 2000. Large Margin DAGs for Multiclass Classification. In: *Advances in Neural Information Processing Systems.* s.l.:MIT Press, pp. 547-553.

Polson, N. & Scott, S., 2011. Data Augmentation for Support Vector Machines. *Bayesian Analysis,* 6(1), pp. 1-23.

Ryman-Tubb, N., Krause, P. & Garn, W., 2018. How Artificial Intelligence and machine learning research impacts payment card fraud detection: A survey and industry benchmark.. *Engineering Application of Artificial Intelligence,* Volume 76, pp. 130-157.

Suvarna, R. & Kowshalya, M., 2020. Credit Card Fraud Detection using Federated Learning Techniques. *International Journal of Scientific research in Science, Engineering and Technology,* 7(3), pp. 356-367.

Weinberger, M., 2022. *Google is turning a key technology into a weapon in its cloud war with Amazon and Microsoft.* [Online]
Available at: https://fortune.com/2015/05/06/google-launches-bigtable-database/
[Accessed 2 January 2023].

Zeng, Y. & Tang, J., 2021. RLC-GNN: An Improved Deep Architecture for Spatial-Based Graph Neural Network with Application to Fraud Detection. *Applied Science,* Volume 11, pp. 56-64.