PDANA8412

Task 2

Contents

[1.1 Logistic regression 3](#_Toc148695412)

[1.2 Objectives 4](#_Toc148695413)

[1.3 Training 4](#_Toc148695414)

[1.4 Advantages and disadvantages 4](#_Toc148695415)

[1.5 Applications 4](#_Toc148695416)

[2.1 The dataset 5](#_Toc148695417)

[3.1 Analysis to be conducted. 5](#_Toc148695418)

[4.1 Data analysis 6](#_Toc148695419)

[4.2 Data cleaning 6](#_Toc148695420)

[4.3 Exploratory data analysis 7](#_Toc148695421)

[4.4 feature engineering 11](#_Toc148695422)

[4.5 Data preprocessing 12](#_Toc148695423)

[4.6 Model building 13](#_Toc148695424)

# 1.1 Logistic regression

Logistic regression is a type of classification in supervised machine learning where its able to conduct specific analysis when the dependant variable is dichotomous (binary), unlike linear regression which is continuous. It’s used to analyse the relationship between one dependent binary variable and one or more nominal, ordinal, interval, or ratio-level independent variables.



Figure 1: Difference between linear and logistic regression

In the figure above shows the difference between linear and logistic regression. In the linear regression graph, it’s unable to fir the data points that are on one extreme sides, because a linear graph is always a straight line. Logistic regression on the other hand is a sigmoid function, S shape, that can fit into the two extreme data points. To achieve this kind of function there is mathematical manipulation done to the linear regression formula:

1. Start with the logistic function:

σ(z) = 1 / (1 + e^(-z))

Here, "z" is any real number.

1. Use the exponential function "e" (approximately equal to 2.71828) raised to the power of negative "z" in the denominator.
2. Combine the terms in the denominator:

σ(z) = 1 / (1 + e^(-z))

1. Calculate the value of e^(-z):

e^(-z) = 1 / e^z

1. Rewrite the expression by multiplying both the numerator and denominator by e^z:

σ(z) = (e^z) / (e^z + 1)

1. Now, we have the logistic sigmoid function in a different form:

σ(z) = (1 + e^(-z))^-1

## 1.2 Objectives

Logistic regression is used to model the relationship between a binary dependent variable (also called the target or outcome variable) and one or more independent variables (predictors or features). The primary goal is to predict the probability of an observation belonging to one of two classes (e.g., 0 or 1, Yes or No, True or False).

## 1.3 Training

The logistic regression model is trained using a dataset where the values of the independent variables and the corresponding class labels are known. The model's parameters (the coefficients) are optimized to best fit the data, typically by maximizing the likelihood of the observed data under the logistic regression model.

## 1.4 Advantages and disadvantages

Advantages:

1. Easy implementation, interpretation, and efficient training.
2. Makes no assumptions about class distributions in feature space.
3. Can extend to multiple classes (multinomial regression) and provides a natural probabilistic view of predictions.
4. Provides predictor importance measures and direction of association.
5. Fast at classifying unknown records.
6. Good accuracy for simple, linearly separable datasets.
7. Less prone to overfitting, with options for regularization (L1 and L2).

Disadvantages:

1. Not suitable when the number of observations is less than the number of features (may lead to overfitting).
2. Constructs linear decision boundaries.
3. Assumes linearity between the dependent variable and independent variables.
4. Can only predict discrete functions due to its binary nature.
5. Inadequate for solving non-linear problems, as it has a linear decision surface.
6. Requires low multicollinearity between independent variables.
7. Limited in capturing complex relationships compared to more advanced algorithms like Neural Networks.

## 1.5 Applications

Logistic regression is a versatile statistical method with numerous applications in various fields. Some of these applications include:

* Credit scoring: ID Finance uses logistic regression for easily interpretable credit scoring models. Data preprocessing includes reducing correlated variables and feature selection. Logistic regression helps identify influential variables and eliminates redundant ones. Results can be exported to Excel for non-technical users to gain insights. Despite exploring Python for model development, logistic regression remains a key component due to its remarkable results in credit scoring.
* Medicine: Miroculus employs logistic regression in medical research to identify links between micro-RNA and genes. Text analysis on scientific articles creates feature vectors for classification. Logistic regression is chosen for its speed and accuracy in predicting binary connections. The algorithm enhances the accuracy of quick blood tests for diseases affected by genes.

# 2.1 The dataset

For this logistic regression, the dataset that is going to be used is the Synthetic Financial Datasets for Fraud Detection dataset. The dataset, which simulates mobile money transactions, could be suitable for logistic regression, particularly for the task of fraud detection. Here are some reasons why this dataset is good for logistic regression: is appropriate for analysis with logistic regression for several reasons:

1. Binary Classification: The dataset includes a binary target variable called "isFraud," which indicates whether a transaction is fraudulent or not. Logistic regression is well-suited for binary classification tasks, making it a good choice to predict and detect fraudulent transactions.
2. Features: The dataset provides several features such as "step" (representing time), "type" of transaction, "amount," "oldbalanceOrg," "newbalanceOrig," "oldbalanceDest," and "newbalanceDest." These features can be used as input variables for logistic regression, making it possible to build a model based on transaction characteristics.
3. Synthetic Data: The dataset is a synthetic representation of real-world financial transactions, generated to resemble normal operations and inject malicious behaviour. This synthetic nature allows researchers to experiment with various fraud detection techniques without privacy concerns and provides a controlled environment for model development.
4. Use Case Relevance: Fraud detection in financial transactions is a common application for logistic regression. The dataset's purpose is to evaluate the performance of fraud detection methods, making it highly relevant for logistic regression modelling in this context.
5. Imbalanced Data: In many fraud detection scenarios, the data is imbalanced, with a majority of non-fraudulent transactions and a small percentage of fraudulent ones. Logistic regression can handle imbalanced datasets well and is suitable for detecting rare events like fraud.
6. Interpretability: Logistic regression models are relatively interpretable, which is important in financial fraud detection. You can easily understand the influence of each feature on the probability of fraud and provide explanations for model predictions.
7. Computational Efficiency: Logistic regression is computationally efficient and can handle large datasets with millions of records, as demonstrated by the scalability of the simulator that generated this dataset.

# 3.1 Analysis to be conducted.

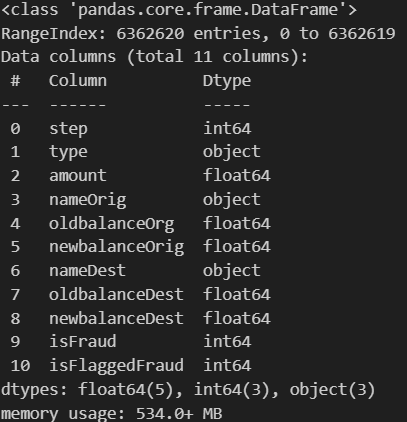
The analysis that will be conducted is a logistic regression model build from scratch via a neural network architecture. This analysis will consist of the following methods:

* Data cleaning: This is where the dataset is initially analysed to see if it must be cleaned of any unwanted variables such as missing or null values, duplications, and imbalances.
* Exploratory data analysis: in this section the dataset will be analysed to have a better understanding of the features and their relationship. This may include a correlation matrix, histogram, pie chart and bar graph. This will be used to see which features have a relationship with the target variable is a transaction being fraudulent or not.
* Feature engineering: This section will change and drop features depending on their relevancy to the classification. In addition, new features maybe developed that may help with the predictability of a fraudulent transaction.
* Data preprocessing: This step involves making sure all redundant features are removed and the data is scaled via normalisation. In addition, techniques like standardization and SMOTE will be applied to categorical variables to be changed into numerical. Then the dataset will be split into training and test sets using the train\_test\_split library.
* Model building: This is where the logistic regression model will be built via a neural net. This will consist of once Dense layer with a binary entropy because a logistic regression only makes a binary outcome in the output layer. This will be run on epochs and the results will be visualised showing the training and testing accuracy vs loss.
* Tuned model: This is where the model will be adjusted to avoid the mistakes of the first model in which overfitting or underfitting could a present. This involves regularisation and balancing the dataset with new weights. The visualisation will be the same as the first model.
* Model evaluation: Boths model are compared in the model evaluation process. This will include metrics like accuracy score, ROC\_AUC score, confusion matrix, and a classification report.
* Model testing: In this last step, the best model will be given new input data and will be seen if it could classify a fraudulent or non-fraudulent transaction.

# 4.1 Data analysis

## 4.2 Data cleaning

A screenshot of a computer

Description automatically generatedThe dataset that was loaded was called fraud.csv as it attains to the synthetic fraud or not fraud dataset that was used for this analysis. The shape of the data is seen by (6362620, 11), meaning that there are 6362620 data points or entries and 11 features in total. There are no presence of null values in each feature and no duplications (0) in the dataset. The features are named oddly, so the names of each feature was renamed to the following:

After this, columns ‘step’ and ‘isFlaggedFruad’ have been dropped due to its unimportance.

## 4.3 Exploratory data analysis

Below is the statistical summary of the numerical variables of the dataset:

A screenshot of a computer

Description automatically generated

A graph of a function

Description automatically generatedThe summary statistics for five variables in this dataset reveal valuable insights about the financial transactions under consideration. The "amount" variable, which represents the transaction amounts, has a wide range, with an average value of approximately $179,861.90 and a significant standard deviation of about $603,858.20. The sender's and receiver's old balances ("sender\_old\_balance" and "receiver\_old\_balance") exhibit similar trends, with average balances of approximately $833,883.10 and $1,100,702.00, respectively, but a notable spread in data, as evidenced by their substantial standard deviations. Furthermore, the "sender\_new\_balance" and "receiver\_new\_balance" variables provide information on the updated balances following transactions, with substantial variability in the data. These statistics offer a comprehensive picture of the data distribution and are crucial for understanding the financial dynamics of the transactions.

The density plot for the amount column shows that the distribution is unimodal and skewed to the right. The peak of the distribution is at approximately 100, and the tail extends to the right. This indicates that there are a few very large amounts in the dataset. The median of the distribution is likely to be lower than the mean, as the mean will be pulled up by the large values. The density plot also shows that there is a small bump at around 0. This could be due to several factors, such as refunds or discounts.

A red circle with black text

Description automatically generated

The pie chart shows that many transactions are non-fraudulent. The proportion of fraudulent transactions is very small, at around 0.1%. This suggests that the fraud detection system is effective in identifying and preventing fraudulent transactions. However, it is important to note that even a small proportion of fraudulent transactions can represent a significant financial loss. Therefore, it is important to continue to monitor the fraud detection system and make improvements as needed. In addition, it is important to be aware of the types of fraud that are most common. This information can be used to target fraud prevention efforts. For example, if the most common type of fraud is chargebacks, then it may be helpful to focus on educating customers about how to avoid chargebacks.

A graph of a graph showing a change

Description automatically generated with medium confidence

A graph with red rectangular bars

Description automatically generated with medium confidenceThe histogram of the changes in send and receive balance shows that the distribution is centred around 0, with a slight skew to the right. This indicates that most transactions result in a small change in balance, with a few transactions resulting in larger changes. The width of the distribution indicates that there is a wide range of change in balance amounts. This could be due to several factors, such as the size of the transactions, the type of transactions, and the frequency of transactions. The skew to the right indicates that there are a few transactions that result in a large increase in balance. This could be due to factors such as deposits, refunds, or bonuses. Overall, the histogram suggests that the changes in send and receive balance are relatively small, with a few exceptions. This is likely since most transactions are for relatively small amounts.

The histogram shows the distribution of transaction amounts for fraud and non-fraud transactions in different transaction types. The transaction types are:

* Cash In
* Cash Out
* Transfer
* Debit
* Credit

For each transaction type, the histogram shows the number of transactions in each amount range. The amount ranges are shown on the x-axis, and the number of transactions is shown on the y-axis. The histograms show that the distribution of transaction amounts is different for fraud and non-fraud transactions. For example, for cash in transactions, most non-fraud transactions are for small amounts, while most fraud transactions are for larger amounts. The histograms also show that the distribution of transaction amounts is different for different transaction types. For example, most cash in transactions are for small amounts, while most cash out transactions are for larger amounts. These observations can be used to inform fraud detection efforts. For example, if a cash in transaction is for a large amount, it may be more likely to be fraudulent.

A screenshot of a graph

Description automatically generated

The provided correlation matrix offers several noteworthy insights into the relationships between the variables. First, there is a positive but relatively weak correlation (0.076688) between the "isfraud" variable, which denotes fraudulent transactions, and the transaction "amount." This suggests that larger transaction amounts are slightly more likely to be associated with fraudulent activities. Second, a moderate positive correlation (0.459304) exists between the "amount" and "sender\_new\_balance" variables, indicating that larger transaction amounts are associated with higher sender new balances. Additionally, strong positive correlations near 1 are observed between "receiver\_old\_balance" and "receiver\_new\_balance" (0.976569) and between "sender\_old\_balance" and "sender\_new\_balance" (0.998803). These correlations reflect the expected relationships between old and new balances in transactions.

Once the EDA is done, columns 'origin' and 'destination' are dropped.

## 4.4 feature engineering

A new feature is developed that is useful when deploying the model. For this, the new feature will be named ‘type2’. This would include the connections between CC (customer and customer), CM (customer and merchant), MC (merchant and customer), and MM (merchant and merchant). We then check the fraud vs valid transactions per type2:

Number of fraud transactions according to type are below:

type2

CC 8213

Name: count, dtype: int64

Number of valid transactions according to type are below:

type2

CC 4202912

CM 2151495

Name: count, dtype: int64

The number of Fraud Transactions in total were 8213 and were made from Customer to Customer. The number of Valid Transactions made from Customer to Customer are 4202912. The number of Valid Transactions made from Customer to Merchant are 2151495.

A comparison of a graph

Description automatically generated

The bar chart shows that the number of fraud transactions is much lower than the number of valid transactions for all transaction types. For example, there are about 100 fraud cash in transactions for every 10,000 valid cash in transactions. The bar chart also shows that the proportion of fraud transactions is highest for cash in and cash out transactions. For these transaction types, the proportion of fraud transactions is about 1%. For the other transaction types, the proportion of fraud transactions is much lower. These observations suggest that fraudsters are more likely to target cash in and cash out transactions. This could be because these transaction types are more difficult to trace.

A graph with red squares

Description automatically generated

The bar chart below shows that the transactions between CC (customer to customer) contains more fraudulent transactions as compared to CM (customer to merchant).

## 4.5 Data preprocessing

This part contains 4 steps. Namely, converting categorical data into numerical (onehot encoding), defining dependant and independent variables, splitting the data into training and testing set, and lastly scaling the data via StandardScaler.

Onehot Encoding was applied to type and type2 columns:

A screen shot of a black screen

Description automatically generated

The data is split into X and y variables for model training:

X = df.drop('isfraud', axis = 1)

y=df['isfraud']

Then the data is split training and testing set:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.30, stratify = df['isfraud'])

Below is the shape of the test and train set:

((4453834, 14), (1908786, 14))

Lastly, the train and test are scaled using standard scaler.

## 4.6 Model building

As mentioned, the model that is needing to be built is a logistic regression from scratch, in other words, without using a skitlearn library. This will be done by constructing an artificial neural network (ANN) with one dense layer:

**Model architecture**

#building first neural network model

epochs = 10

model = Sequential()

model.add(Dense(1, input\_dim=X\_train.shape[1], activation='sigmoid'))

* **Epoch = 10**: this is used to create an epoch variable with 10 iterations.
* **model = Sequential():** This creates an empty sequential model, which is a linear stack of layers.
* **model.add(Dense(1, input\_dim=X\_train.shape[1], activation='sigmoid')**: This adds a single Dense layer to the model. The Dense layer represents a fully connected layer in a neural network. It has one output unit (1) and uses the sigmoid activation function. The **input\_dim** parameter is set to the number of features in the input data (**X\_train.shape[1]**), which specifies the number of input neurons.

**Model Compilation**

model.compile(optimizer='Adam', loss = 'binary\_crossentropy', metrics=['accuracy'])

* **model.compile(...)**: This step configures the model for training.
* **optimizer='Adam'**: It uses the Adam optimization algorithm, which is a popular optimizer for gradient-based optimization in neural networks.
* **loss='binary\_crossentropy'**: This specifies the loss function to be used for binary classification. Binary cross-entropy (or log loss) is commonly used for logistic regression and binary classification tasks.
* **metrics=['accuracy']**: During training, it tracks and displays the classification accuracy as a metric.

**Model training**

history = model.fit(np.array(X\_train), np.array(y\_train), epochs=epochs, validation\_data=(np.array(X\_test), np.array(y\_test)), batch\_size=64)

* **model.fit(...):** This method trains the model on the training data.
* **X\_train and y\_train** are the training features and labels, respectively.
* **epochs**: This parameter specifies the number of training epochs (iterations over the dataset).
* **validation\_data=(X\_test, y\_test**): It allows you to specify validation data, which is used to monitor the model's performance during training.
* **batch\_size=64**: This parameter determines the number of samples used in each forward and backward pass during each training iteration. It's useful for mini-batch gradient descent.

The code builds a simple logistic regression model as a single-layer neural network, compiles it with the Adam optimizer and binary cross-entropy loss, and then trains the model on the provided training data. The purpose of this code is binary classification, which is often a task related to logistic regression.

A graph with a line

Description automatically generatedA graph with a line graph

Description automatically generated

The training results for the model show that it achieves a high level of accuracy on both the training and validation datasets. Specifically, the model attains an impressive 99% accuracy on both the training and validation data, which could be perceived as a positive outcome. However, the similarity in accuracy between the training and validation sets (both at 99%) suggests that the model might be overfitting. Overfitting occurs when the model performs exceptionally well on the training data but struggles to generalize to unseen data. Therefore, while the high accuracy is promising, it's essential to scrutinize the model's performance further, including its ability to generalize to new, unseen data, and consider potential steps to address overfitting.

## 4.7 Tuned model

In this tuned model, adjustments were made to original model to combat overfitting. This includes L1 and L2 regularisation and weighted averages. The regularisation will add penalty term to the model’s coefficient to stop it from overfitting. Weighted averages are them to balance the dataset. In the pie chart from EDA, the non-fraud was 99% whilst the fraud was 0.1%. Therefore, this weighted average is used to balance the scale of the ‘isfraud’ class.

weights\_assigned = {0: 1, 1: 770}

epochs = 10

model = Sequential()

model.add(Dense(1, input\_dim=X\_train.shape[1], activation='sigmoid', kernel\_regularizer=l1\_l2(l1=0.01, l2=0.01)))

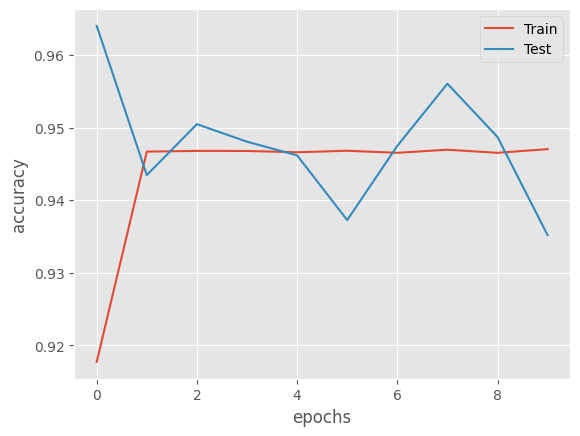
**Class Weights**:

* In the original model, no class weights were specified. In the modified model, you've added class weights using the **class\_weight** parameter. Specifically, you've assigned a weight of 1 to class 0 and a weight of 770 to class 1. This change is made to address class imbalance, where one class (class 1) has significantly fewer samples than the other (class 0).

**Regularization**:

* In the modified model, you've added L1 and L2 regularization to the Dense layer using **kernel\_regularizer**. This means that both L1 (Lasso) and L2 (Ridge) regularization are applied to the weights of the layer. This is intended to prevent overfitting by adding a penalty on the magnitude of the weights.

A graph with red and blue lines

Description automatically generated

For the tuned model, the accuracy is lower than the original model with an accuracy of 95%. The loss of the validation set is lower than the original model from 51% to 30%. This variation of accuracy and loss for both training and validation set shows that overfitting is lowered, and the tuned model performs equally as well with an above 90% accuracy.

## 4.8 Model evaluation

The original model is saved as “logreg\_model.h5” and the tuned model is saved as “logreg\_model2.fixed.”

# Load the first model

model1 = keras.models.load\_model('logreg\_model.h5')

# Load the second model

model2 = keras.models.load\_model('logreg\_model2.fixed')

Here's a comparison of the two models:

1. **Accuracy**:
   * Model 1 has a significantly higher accuracy (99.93%) compared to Model 2 (93.52%). This suggests that Model 1 is better at correctly classifying both classes, but it's important to consider class imbalance and the impact on overall accuracy.
2. **Precision and Recall**:
   * Model 1 has a higher precision for class 1 (fraud), indicating that when it predicts fraud, it's more likely to be correct. However, its recall (true positive rate) for class 1 is relatively low (46%).
   * Model 2, on the other hand, has a much higher recall for class 1 (91%), but its precision is very low.
3. **F1-Score**:
   * The F1-score, which balances precision and recall, is better for class 1 in Model 1 (0.62) but is very low in Model 2 (0.03).
4. **ROC AUC Score**:
   * Model 2 has a significantly higher ROC AUC score (0.921) compared to Model 1 (0.732). The ROC AUC score indicates that Model 2 is better at distinguishing between the two classes, particularly with class 1 (fraud).
5. **Confusion Matrix**:
   * Model 1 has fewer false positives but a higher number of false negatives.
   * Model 2 has a significantly higher number of false positives but a lower number of false negatives.

Overall, model2 is better to be used in a logistic regression classifying a fraudulent transaction than model1 and this is because the relative accuracy of both models is still high at above 90% and in addition to the fact that model1 had to be balanced considering the overwhelming non-fraudulent transactions. In addition, the ROC AUC score is better in model2, meaning that its better at distinguishing between the two classes of fraud and non-fraud.

## 4.9 Testing model

The model that was tested is the model2 of the logreg\_model\_fixed. New data was given manually so that anyone else can adjust the new data. Here’s the results:

1/1 [==============================] - 0s 21ms/step

Fraud Probability: [[1.]]

Predicted Class (1=Fraud, 0=Not Fraud): [[1]]

Therefore, with the new data, the model predicted a fraudulent case.

# 5 Reference

EDGAR LOPEZ-ROJAS, ELR. (2016) ‘Synthetic Financial Datasets For Fraud Detection’. Available at: <https://www.kaggle.com/datasets/ealaxi/paysim1> (Accessed 17/10/2023)