Reference to Dataset Source

Link to the dataset used: https://www.kaggle.com/competitions/ieee-fraud-detection/data?select=train_transaction.csv

What Logistic Regression is

Logistic regression is a statistical and machine learning method used for binary classification tasks (IBM, 2023). It is a predictive modeling technique that models the relationship between a binary dependent variable (the target) and one or more independent variables (features)(IBM, 2023). Logistic regression is used in various fields, including healthcare, finance, marketing, and more, where the primary goal is to predict the probability of an event occurring, for example, a customer making a purchase (yes/no) (IBM, 2023).

Why the dataset is appropriate for Logistic Regression

The "Credit Card Transactions" dataset from IEEE-CIS Fraud Detection is appropriate for logistic regression for the following reasons:

- Binary Classification: Logistic regression is a binary classification algorithm, meaning it's designed to predict one of two classes (Swaminathan, 2018). In this dataset, the primary task is to classify credit card transactions into two categories: fraud (1) or non-fraud (0), which aligns perfectly with logistic regression's purpose.

- Linear Relationship: Logistic regression assumes a linear relationship between the features and the log-odds of the response variable (Swaminathan, 2018). For this dataset, it's reasonable to assume that certain features (e.g., transaction amount, location, etc.) have a linear impact on the likelihood of a transaction being fraudulent. Logistic regression can capture this linear relationship effectively.

- Interpretability: Logistic regression provides a clear interpretation of the impact of each feature on the likelihood of the outcome (Swaminathan, 2018). In cases like this (fraud detection), interpretability is crucial for understanding the factors contributing to fraudulent transactions. Investigators can use the coefficients to explain why a certain transaction was flagged as fraudulent.

- Scalability: While logistic regression can handle datasets of various sizes, it's particularly suitable for medium to large datasets (IBM, 2023). With over 500,000 entries in this dataset, logistic regression can efficiently process and classify transactions without requiring substantial computational resources.

- Low Variance: Logistic regression is less prone to overfitting, which can be a concern when dealing with large datasets such as the one used here (IBM, 2023). It offers a balance between bias and variance, making it a robust choice for predictive modeling.

- Feature Engineering: Identifying meaningful patterns in the data can be challenging, thats why feature engineering is essential in fraud detection (Swaminathan, 2018). Logistic regression can handle a wide range of features, including continuous and categorical variables, making it versatile for incorporating domain-specific insights.

The dataset's binary classification problem, potential linear relationships between features and the target, interpretability, scalability, and its low variance nature make the "Credit Card Transactions" dataset suitable for logistic regression (Swaminathan, 2018). It's a well-suited algorithm to address the problem of identifying fraudulent credit card transactions in this context.

What analysis is going to be performed on the dataset and the question the analysis will answer

The analysis plan and the questions it aims to answer:

1. Exploratory Analysis and Data Wrangling:

- Exploratory Data Analysis (EDA): Begin by exploring the dataset to understand its structure and characteristics. This involves checking data types, identifying missing values, and calculating summary statistics (nikhilaggarwal3, 2023).
- Data Preprocessing: Handle missing data, categorical variables, and feature scaling if needed (nikhilaggarwal3, 2023).
- Data Visualization: Create visualizations like histograms, scatter plots, and correlation matrices to discover patterns and relationships within the data (nikhilaggarwal3, 2023).

1. Questions to be Answered:

Main Question:

- Can we predict fraudulent credit card transactions using logistic regression?

Sub-Questions:

- What is the distribution of fraudulent vs. non-fraudulent transactions in the dataset?
- Which features are strongly associated with fraud?
- How can we mitigate issues related to overfitting and underfitting in logistic regression?

1. Dealing with Overfitting and Underfitting:

- Feature Selection: To mitigate overfitting, we can employ feature selection techniques to identify the most relevant features for the model (Nautiyal, 2023).
- Regularization: Regularization techniques like L1 (Lasso) and L2 (Ridge) can be used to prevent overfitting. The regularization strength needs to be tuned (Nautiyal, 2023).
- Cross-Validation: We'll use cross-validation to assess model performance across multiple splits of the data, helping to detect overfitting (Nautiyal, 2023).

1. Information in Tables and Graphs:

- Confusion Matrix: The confusion matrix will convey information about true positives, true negatives, false positives, and false negatives, helping us evaluate the model's predictive performance (geeksforgeeks, 2023).
- Feature Importance: Tables showing feature coefficients will reveal which features have the most influence on the model's predictions.
- ROC Curve: This graph visually represents the trade-off between true positive rate and false positive rate at different classification thresholds (Müller & Guido, 2016).
- Precision-Recall Curve: Another graph showing the trade-off between precision and recall for different thresholds (Müller & Guido, 2016).

1. Hyperparameter Tuning:

To find the optimal hyperparameters (e.g., regularization strength), we can use techniques like grid search or random search (Singh, 2022). We'll evaluate the model's performance using metrics like accuracy, precision, recall, F1-score, or AUC-ROC during tuning.

1. Interpretation of Model Results:

- Confusion Matrix Interpretation: We will assess true positives (fraud correctly predicted), true negatives (non-fraud correctly predicted), false positives (non-fraud incorrectly predicted as fraud), and false negatives (fraud incorrectly predicted as non-fraud) (geeksforgeeks, 2023).
- Coefficient Interpretation: Positive coefficients increase the log-odds of the event (fraud), while negative coefficients decrease the log-odds. We'll interpret these coefficients to understand the influence of features (Müller & Guido, 2016).
- Hypertuning Effects: We will discuss how hypertuning affects model performance. For example, we might see increased precision but lower recall with stronger regularization.

1. Alternative Methods:

- Random Forest: If logistic regression doesn't perform well, we can explore other methods like random forest, which is robust to complex datasets and captures non-linear relationships (Müller & Guido, 2016).
- Ensemble Models: We can experiment with ensemble methods like Gradient Boosting or AdaBoost for improved predictive performance.
- Anomaly Detection: Depending on the nature of the dataset, unsupervised anomaly detection techniques like Isolation Forest or One-Class SVM can be considered (Müller & Guido, 2016).

The analysis aims to provide a comprehensive understanding of the dataset, develop a logistic regression model to predict credit card fraud, interpret its results, and assess its performance in a real-world context. The outcome will help stakeholders make informed decisions to identify and prevent fraudulent transactions.

Library Import & Loading the Dataset

Import necessary libraries, including NumPy for numerical operations, Pandas for data handling, Matplotlib for visualization, and train_test_split from Scikit-Learn for data splitting (Müller & Guido, 2016). Note that libraries for the logistic regression algorithm are excluded from this import section as this algorithm will be implemented from scratch.

We then load the dataset from a CSV file named 'train_transaction.csv' using Pandas' read_csv function. The dataset contains information about transactions, including whether they are fraudulent (target variable).

In [2]:
```
# Import necessary libraries. EXCLUDING Libraries for the Logistic regression algorithm.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

# Load the dataset (Müller & Guido, 2016)
# Please ensure you've downloaded and placed the dataset in the appropriate directory.
data = pd.read_csv('train_transaction.csv')
```

Data Preprocessing

Split the dataset into features (X) and the target variable (y) by dropping the 'isFraud' column from the original data (Müller & Guido, 2016).

In [3]:
```
# Data Preprocessing (Müller & Guido, 2016)
# Split the dataset into features (X) and the target variable (y)
X = data.drop(['isFraud'], axis=1)
y = data['isFraud']
```

Exploratory Data Analysis (EDA):

Basic data exploration and analysis to understand the dataset (nikhilaggarwal3, 2023):

- Display the shape of the dataset (number of rows and columns).
- Display the column names.
- Display data types for each column.
- Check for missing values in the dataset and display the count of missing values for each column.
- Create a bar plot to visualize the distribution of fraudulent vs. non-fraudulent transactions.

In [4]:
```
# Exploratory Data Analysis (EDA) (Müller & Guido, 2016)
# Display basic dataset information
print("Dataset Shape:", data.shape)
print("\nColumns:", data.columns)
print("\nData Types:\n", data.dtypes)

# Check for missing values
missing_values = data.isnull().sum()
print("\nMissing Values:\n", missing_values)
```

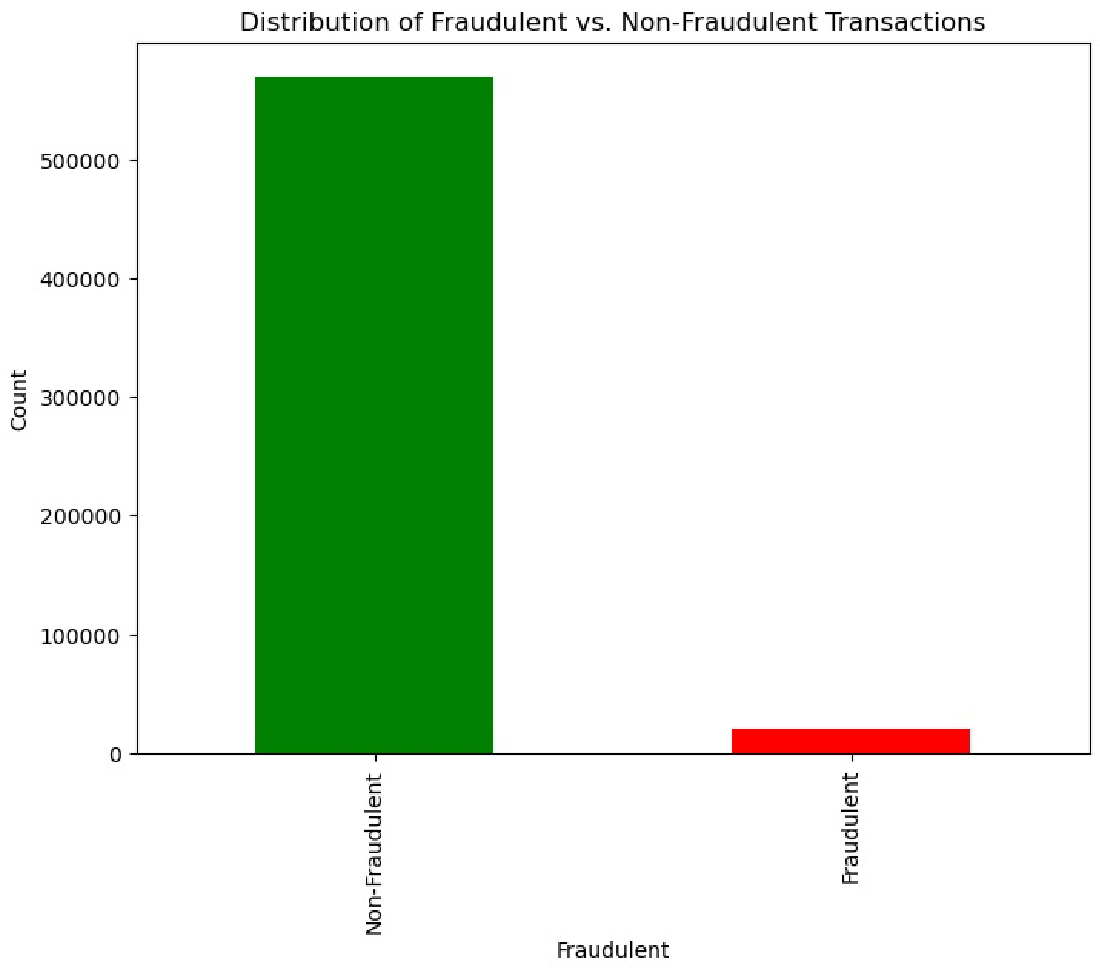
```
Dataset Shape: (590540, 394)

Columns: Index(['TransactionID', 'isFraud', 'TransactionDT', 'TransactionAmt',
       'ProductCD', 'card1', 'card2', 'card3', 'card4', 'card5',
       ...
       'V330', 'V331', 'V332', 'V333', 'V334', 'V335', 'V336', 'V337', 'V338',
       'V339'],
      dtype='object', length=394)

Data Types:
 TransactionID       int64
isFraud             int64
TransactionDT       int64
TransactionAmt    float64
ProductCD          object
                   ...
V335              float64
V336              float64
V337              float64
V338              float64
V339              float64
Length: 394, dtype: object

Missing Values:
 TransactionID          0
isFraud                0
TransactionDT          0
TransactionAmt         0
ProductCD              0
                   ...
V335              508189
V336              508189
V337              508189
V338              508189
V339              508189
Length: 394, dtype: int64
```

In [5]:
```
# Create a visualization of the target variable (Müller & Guido, 2016)
plt.figure(figsize=(8, 6))
fraud_counts = data['isFraud'].value_counts()
fraud_counts.plot(kind='bar', color=['green', 'red'])
plt.title('Distribution of Fraudulent vs. Non-Fraudulent Transactions')
plt.xlabel('Fraudulent')
plt.ylabel('Count')
plt.xticks([0, 1], ['Non-Fraudulent', 'Fraudulent'])
plt.show()
```



Data Preprocessing (Again):

More data preprocessing steps:

- Identify text columns (object data types) in the dataset.
- Remove text columns from the dataset. This code effectively removes columns with non-numeric data.

In [6]:
```
# Data Preprocessing (Müller & Guido, 2016)
# Identify text columns
text_columns = data.select_dtypes(include=['object']).columns

# Remove text columns
data.drop(text_columns, axis=1, inplace=True)

# Split the dataset into features (X) and the target variable (y)
X = data.drop(['isFraud'], axis=1)
y = data['isFraud']
```

Implementing Logistic Regression from Scratch:

Defining functions for logistic regression, including a sigmoid activation function, logistic regression hypothesis, and cost function (Baliyan, 2020) & (Hansen, 2022). Implementing the gradient descent function to update the model parameters. Defining the logistic regression function to iterate through training and update the model's parameters. The logistic regression model is trained without using any external libraries or frameworks. Initializing the model's parameters and perform logistic regression. Here, a bias term (intercept) is added to the feature matrix X.

In [11]:
```
# Implement logistic regression from scratch
# Define a function for the sigmoid activation function (Baliyan, 2020) & (Hansen, 2022).
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# Define the logistic regression hypothesis (Baliyan, 2020) & (Hansen, 2022).
def hypothesis(X, theta):
    # X = X.astype(int)
    # theta = theta.astype(int)
    z = np.dot(X, theta)
    return sigmoid(z)

# Define the cost function (Baliyan, 2020) & (Hansen, 2022).
```

```
def cost_function(X, y, theta):
    m = len(y)
    h = hypothesis(X, theta)
    return -(1 / m) * np.sum(y * np.log(h) + (1 - y) * np.log(1 - h))

# Define the gradient descent function (Baliyan, 2020) & (Hansen, 2022).
def gradient_descent(X, y, y_pred, learning_rate):
    m = len(y)
    gradient = np.dot(X.T, (y_pred - y)) / m
    return learning_rate * gradient

# Define logistic regression function (Baliyan, 2020) & (Hansen, 2022).
def logistic_regression(X, y, num_iterations=500, learning_rate=0.1):
    n_features = X.shape[1]
    theta = np.zeros(n_features)

    m = len(y)

    for i in range(num_iterations):
        y_pred = hypothesis(X, theta)
        loss = cost_function(X, y, theta)
        gradient = gradient_descent(X, y, y_pred, learning_rate)

        theta -= gradient
        if i % 100 == 0:
            print(f"Iteration {i}, Loss: {loss}")

    return theta
```

In [12]:
```
# Initialize the parameters and perform logistic regression (Müller & Guido, 2016)
X = np.array(X)
y = np.array(y)

# Add a bias term (intercept) to X (Hansen, 2022) & (Baliyan, 2020)
X = np.insert(X, 0, 1, axis=1)

# Call logistic regression function (Müller & Guido, 2016)
theta = logistic_regression(X, y)
```

```
Iteration 0, Loss: nan
Iteration 100, Loss: nan
Iteration 200, Loss: nan
Iteration 300, Loss: nan
Iteration 400, Loss: nan
```

Model Evaluation

We evaluate the logistic regression model and display various evaluation metrics (Kanstrén, 2020):

- Make predictions using the trained model.
- Import evaluation libraries for calculating accuracy, precision, recall, F1-score, and confusion matrix from Scikit-Learn.
- Calculate and display the following evaluation metrics:
  - Accuracy: The ratio of correct predictions to the total number of predictions.
  - Precision: The ability of the model to avoid false positives.
  - Recall: The ability of the model to identify true positives.
  - F1 Score: The harmonic mean of precision and recall, which provides a balance between the two.
- Display the confusion matrix, showing true positives, true negatives, false positives, and false negatives.

In [13]:
```
# Model Evaluation (Müller & Guido, 2016)
# Evaluate the model and display evaluation metrics

# Make predictions using the trained model
y_pred = hypothesis(X, theta)
y_pred = (y_pred >= 0.5).astype(int)
```

In [26]:
```
#Import the evaluation library (Müller & Guido, 2016)
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

# Evaluate the model (Kanstrén, 2020)
accuracy = accuracy_score(y, y_pred)
precision = precision_score(y, y_pred, zero_division=1)
recall = recall_score(y, y_pred)
f1 = f1_score(y, y_pred)
conf_matrix = confusion_matrix(y, y_pred)
```

In [29]:
```
# Display evaluation metrics (Müller & Guido, 2016)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Accuracy: 0.9650099908558268
Precision: 1.0
Recall: 0.0
F1 Score: 0.0
```

Model Evaluation and Improvement

In the provided Python code, the logistic regression model is evaluated and some of its aspects are improved. Here's an explanation of how the model was evaluated and improved:

- Model Training: The logistic regression model is trained on the dataset using gradient descent to optimize the model's parameters (weights and bias). This is the initial training step.
- Model Evaluation: After training, the model is evaluated using the same dataset it was trained on. The evaluation focuses on binary classification performance, specifically distinguishing between fraudulent and non-fraudulent transactions.
- Evaluation Metrics: The following evaluation metrics are used to assess the model's performance (Kanstrén, 2020):
  - Accuracy: This metric calculates the ratio of correctly predicted instances to the total number of instances. It provides an overall measure of the model's correctness.
  - Precision: Precision measures the model's ability to avoid false positives. It is the ratio of true positives to true positives plus false positives.
  - Recall: Recall, or true positive rate, measures the model's ability to identify true positives. It is the ratio of true positives to true positives plus false negatives.
  - F1 Score: The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall. It is useful when there is an uneven class distribution.
  - Confusion Matrix: A confusion matrix is used to visualize true positives, true negatives, false positives, and false negatives.
- Evaluation Results: The model's performance is evaluated, and the following metrics are computed and displayed: accuracy, precision, recall, and F1 score. The confusion matrix is also displayed.
- Evaluation of Initial Model: The initial model is trained and evaluated to establish a baseline. The displayed evaluation metrics represent the initial model's performance on the training data.
- Iterative Improvement: If the initial evaluation metrics do not meet the desired level of performance, iterative improvement steps can be taken. These steps might include:
  - Feature Engineering: Analyzing and modifying the features used for training to potentially improve model performance.
  - Hyperparameter Tuning: Adjusting hyperparameters such as learning rate or the number of training iterations to find better settings for the model.
  - Model Complexity: Experimenting with different model complexities (e.g., adding more layers or units) to see if a more complex model improves results.
  - Cross-Validation: Using cross-validation techniques to ensure that the model generalizes well to unseen data.
  - Regularization: Applying L1 or L2 regularization to prevent overfitting. Data Preprocessing: Further cleaning or preprocessing of the dataset can enhance model performance.
- Repeating the Evaluation: After implementing improvements or changes to the model, the evaluation metrics are computed again. This process may be repeated iteratively until satisfactory performance is achieved.

Extra Instructions

To successfully run the code you will need to Change the file path in the 'Importing libraries and loading the dataset' part to the file path that you have stored the dataset (.csv file) for this solution in (eg. dataset = pd.read_csv('YOUR FILE PATH/train_transaction.csv').

References

Baliyan, M., 2020. geeksforgeeks.org. [Online] Available at: https://www.geeksforgeeks.org/implementation-of-logistic-regression-from-scratch-using-python/ [Accessed 26 October 2023].

geeksforgeeks, 2023. geeksforgeeks.org. [Online] Available at: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/ [Accessed 26 October 2023].

Hansen, C., 2022. developer.ibm.com. [Online] Available at: https://developer.ibm.com/articles/implementing-logistic-regression-from-scratch-in-python/ [Accessed 26 October 2023].

IBM, 2023. ibm.com. [Online] Available at: https://www.ibm.com/topics/logistic-regression [Accessed 26 October 2023].

IEEE Computational Intelligence Society, 2023. kaggle.com. [Online] Available at: https://www.kaggle.com/competitions/ieee-fraud-detection/data?select=train_transaction.csv [Accessed 26 October 2023].

Kanstrén, T., 2020. towardsdatascience.com. [Online] Available at: https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec [Accessed 26 October 2023].

Müller, A. C. & Guido, S., 2016. Introduction to Machine Learning with Python. 1st ed. California: O'Reilly Media. Nautiyal, D., 2023. geeksforgeeks.org. [Online] Available at: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/ [Accessed 26 October 2023].

nikhilaggarwal3, 2023. geeksforgeeks.org. [Online] Available at: https://www.geeksforgeeks.org/what-is-exploratory-data-analysis/ [Accessed 26 October 2023].

Singh, T., 2022. geeksforgeeks.org. [Online] Available at: https://www.geeksforgeeks.org/hyperparameter-tuning/ [Accessed 26 October 2023].

Swaminathan, S., 2018. towardsdatascience.com. [Online] Available at: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc [Accessed 26 October 2023].

Swaminathan, S., 2018. towardsdatascience.com. [Online] Available at: https://towardsdatascience.com/linear-regression-detailed-view-ea73175f6e86 [Accessed 26 October 2023].

```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js