CREDIT CARD FRAUD DETECTION – ML ASSIGNEMENT

Machine Learning - IT4060

Sudhais F.M

IT21098000

B.Sc. (Hons) Degree in Information Technology specialized in

Software Engineering

Department of Software Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

April 2025

Contents

1. Introduction

Credit card fraud has emerged as a significant threat in the financial sector, costing billions of dollars annually to businesses, banks, and consumers worldwide. With the rapid growth of online transactions and digital payment systems, fraudsters have found increasingly sophisticated ways to exploit vulnerabilities, making fraud detection more critical than ever. Timely and accurate identification of fraudulent credit card transactions is essential to minimize financial loss and maintain customer trust.

Credit card fraud detection presents a unique challenge due to the highly imbalanced nature of transaction data fraudulent transactions typically account for a very small percentage of the total, while genuine transactions dominate the dataset. Traditional manual review methods are inefficient and cannot scale with the growing volume of transactions, thus necessitating the use of automated, data-driven approaches.

Logistic regression, a supervised machine learning algorithm, offers a robust and interpretable solution for binary classification problems like fraud detection. It predicts the probability that a given transaction is fraudulent based on a set of input features such as transaction amount, time, and user behavior patterns. Despite being a relatively simple model, logistic regression is widely used due to its computational efficiency, transparency, and ability to perform well in many real-world scenarios when appropriately tuned and combined with data preprocessing techniques.

In this report, we investigate the application of logistic regression for detecting fraudulent credit card transactions using a publicly available dataset. The goal is to build a model that can distinguish between legitimate and fraudulent transactions by learning patterns from historical data. We also explore the challenges associated with imbalanced datasets and discuss strategies used to address them, including sampling methods and performance metrics beyond accuracy.

This work not only demonstrates the effectiveness of logistic regression in identifying fraud but also highlights the importance of careful data preprocessing, feature selection, and evaluation metrics in developing reliable fraud detection systems

2. Problem Addressed

In recent years, the use of credit cards has significantly increased, leading to a parallel rise in fraudulent activities. Credit card fraud causes substantial financial losses to individuals, merchants, and financial institutions. Detecting fraudulent transactions is a challenging task due to several key factors:

- The **imbalance of data**, where fraudulent transactions represent a very small percentage of all transactions.
- The **evolving nature of fraud patterns**, which makes traditional rule-based systems ineffective over time.
- The **need for real-time or near-real-time detection**, where delays in identifying fraud can result in greater financial damage.

The primary problem addressed in this study is the **accurate detection of credit card fraud** using **supervised machine learning techniques**, specifically through the use of **Logistic Regression**. The goal is to build a model that can learn from historical transaction data and correctly classify new transactions as either legitimate or fraudulent, with a strong focus on **high recall** to minimize the number of undetected frauds.

This project explores:

- How to handle imbalanced datasets effectively.
- The performance of a basic yet interpretable machine learning algorithm (Logistic Regression) for binary classification.
- The trade-offs between accuracy, precision, and recall in fraud detection models

3. Dataset Used

Source and Overview

The dataset used in this project is publicly available from Kaggle and originates from research conducted by European credit card issuers. It contains anonymized data for transactions that took place over a period of two days in September 2013.

Characteristics of the Dataset

- **Total Transactions**: 284,807
- **Fraudulent Transactions**: 492 (0.172%)
- **Non-Fraudulent Transactions**: 284,315
- **Number of Features**: 30
    - V1 through V28: Principal components obtained via PCA
    - Time: Seconds elapsed between each transaction and the first transaction
    - Amount: Transaction amount
    - Class: Binary label (0 = legitimate, 1 = fraud)

Data Challenges

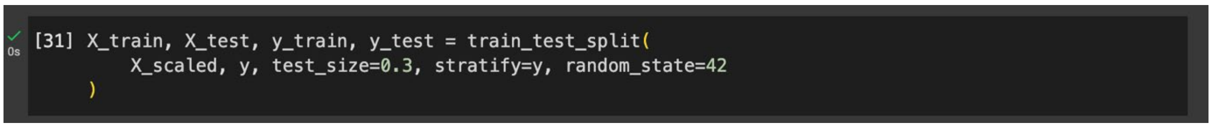
The primary challenge of this dataset lies in its **high class imbalance** — fraudulent transactions comprise less than 0.2% of the total records. Traditional classifiers may achieve high accuracy simply by predicting all transactions as non-fraudulent. Hence, using appropriate evaluation metrics and class-weighting strategies is crucial.

4. Methodology

Data Preprocessing

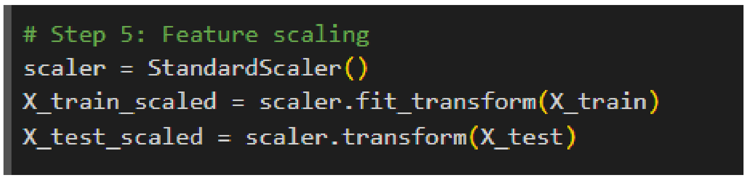
Several preprocessing steps were applied:

1. **Feature and Label Separation**:
   - Input features (X) include all columns except the Class.
   - Output label (y) is the Class column.
2. **Train-Test Split**:
   - Dataset was split into **80% training** and **20% testing** using stratified sampling to maintain class proportions.



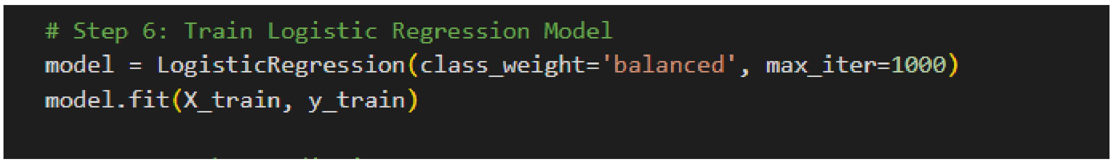
```
[31] X_train, X_test, y_train, y_test = train_test_split(
         X_scaled, y, test_size=0.3, stratify=y, random_state=42
     )
```

3. **Feature Scaling**:
   - All features were scaled using **StandardScaler** to normalize values, especially important for Logistic Regression.
   - Although the PCA-transformed features (V1–V28) are already standardized, the Amount and Time features were not. We applied **standard scaling** to these features using StandardScaler to normalize their values.



```
# Step 5: Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

4. **Handling Class Imbalance**:
   - Used **class_weight = 'balanced'** in the logistic regression model to give more weight to the minority class (fraud cases).



```
# Step 6: Train Logistic Regression Model
model = LogisticRegression(class_weight='balanced', max_iter=1000)
model.fit(X_train, y_train)
```

Model Selection

Logistic Regression, a widely used linear classifier for binary classification problems, was selected for its simplicity, interpretability, and effectiveness in handling linearly separable data.

Model parameters:

- Solver: default (lbfgs)
- Class Weight: 'balanced' to account for data imbalance
- Random State: 42 (for reproducibility)

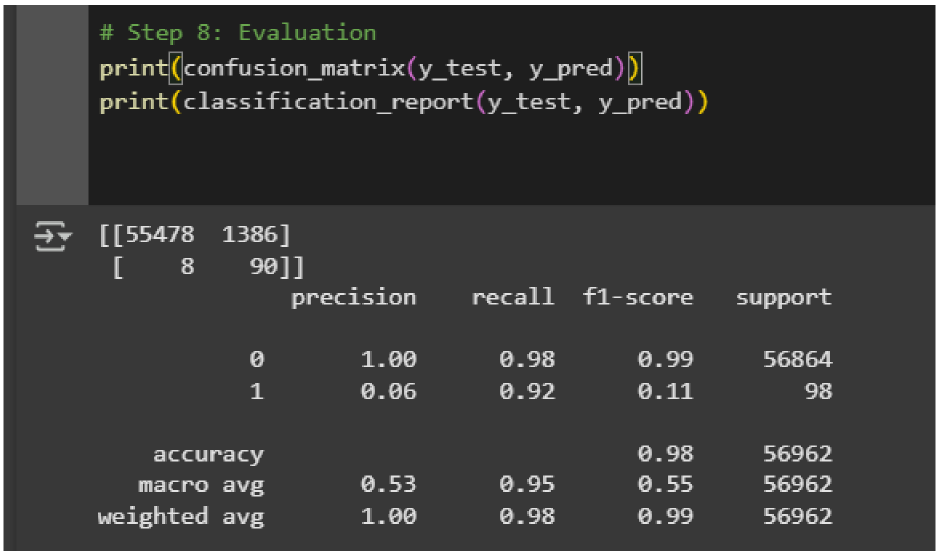
Model Training

The model was trained on the scaled training dataset. It learned to distinguish between fraudulent and legitimate transactions using the labeled data.

5. Evaluation Metrics

To evaluate the model, the following metrics were used:

- **Confusion Matrix**: To visualize the counts of true/false positives and negatives.
- **Precision**: The ratio of true fraud predictions to all predicted frauds.
- **Recall**: The ratio of true fraud predictions to all actual frauds (also known as Sensitivity).
- **F1-Score**: Harmonic mean of precision and recall.
- **Accuracy**: Overall correctness of the model (less useful with imbalanced data).


```
# Step 8: Evaluation
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[55478  1386]
 [    8    90]]
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     56864
           1       0.06      0.92      0.11        98

    accuracy                           0.98     56962
   macro avg       0.53      0.95      0.55     56962
weighted avg       1.00      0.98      0.99     56962
```

6. Results and Discussion

This section presents the performance of the Logistic Regression model in detecting credit card fraud, based on the test dataset. The evaluation includes both numerical and graphical results to assess the model's effectiveness in handling highly imbalanced data.

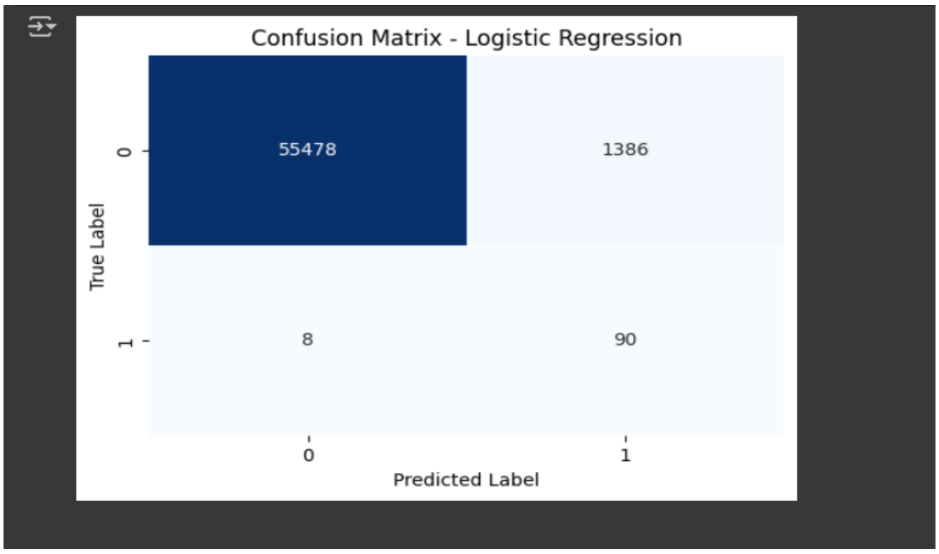
Confusion Matrix:

The confusion matrix showed the following values:

	Predicted Legitimate	**Predicted Fraudulent**
Actual Legitimate	55,478	1,386
Actual Fraudulent	8	90

The confusion matrix shows:

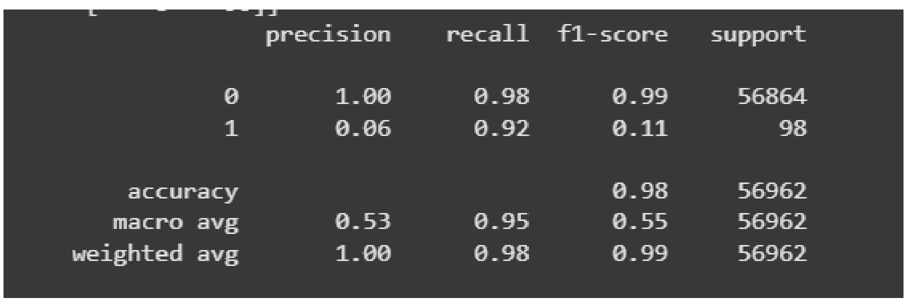
- **True Negatives (TN)**: 55,478 — legitimate transactions correctly identified.
- **False Positives (FP)**: 1,386 — legitimate transactions incorrectly flagged as fraud.
- **False Negatives (FN)**: 8 — fraudulent transactions missed by the model.
- **True Positives (TP)**: 90 — fraudulent transactions correctly detected.



Classification Report:

Class	Precision	Recall	F1-score	Support
Legitimate (0)	1.00	0.98	0.99	56,864
Fraudulent (1)	0.06	0.92	0.11	98

- **Accuracy**: 98%
- **Macro Average F1-score**: 0.55
- **Weighted Average F1-score**: 0.99



```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     56864
           1       0.06      0.92      0.11        98

    accuracy                           0.98     56962
   macro avg       0.53      0.95      0.55     56962
weighted avg       1.00      0.98      0.99     56962
```

Discussion:

Insights from Results

Despite using a simple and interpretable model like **Logistic Regression**, the model performs quite well in detecting fraudulent transactions, especially in terms of **recall** (92%) on the minority class (fraudulent cases). This means the model is able to identify the vast majority of fraudulent transactions — which is critical in real-world scenarios where fraud has high financial impact.

However, the **precision is low (6%)**, meaning that many transactions flagged as fraud are actually legitimate. While this could be inconvenient in some settings (e.g., user experience), it's often acceptable in fraud detection, where **catching actual fraud is more important than occasional false alarms**.

This result reflects a common trade-off in fraud detection systems: **high recall but low precision**. In production systems, additional layers (manual review, heuristics, or ensemble models) are often used to reduce false positives further.

7. Limitations and Future Work

Limitations

While the Logistic Regression model achieved high recall in detecting fraudulent transactions, the study faces several limitations:

- **Class Imbalance**: The dataset is highly imbalanced, with fraud cases representing less than 0.2% of all transactions. Although class weights were adjusted, logistic regression alone may not be the most effective model in handling extreme imbalance.
- **Low Precision**: The model flagged many legitimate transactions as fraudulent, resulting in a **low precision (6%)**. In practical systems, this may lead to unnecessary alerts or customer dissatisfaction.
- **Limited Feature Interpretability**: Most input features were anonymized using PCA (V1 to V28), which makes it difficult to understand which specific transaction characteristics drive fraud predictions.
- **Assumption of Linearity**: Logistic regression assumes a linear relationship between features and the log-odds of the outcome, which may oversimplify the complex patterns involved in fraudulent activities.
- **No Time-Series or Behavioral Analysis**: The model does not consider time-based behavior or user transaction history, which are important for detecting evolving fraud patterns.

Future Work:

To address these limitations and further improve model performance, the following future directions are recommended:

- **Advanced Algorithms**: Implement more complex models such as **XGBoost**, **Random Forest**, **LightGBM**, or **Neural Networks**, which can capture nonlinear relationships and interactions between features.
- **Anomaly Detection Techniques**: Use **unsupervised learning** or **semi-supervised models** to detect outliers without needing labeled data, useful when fraudulent labels are scarce or evolving.
- **Ensemble Methods**: Combine multiple classifiers using techniques like **bagging** or **boosting** to increase robustness and improve both precision and recall.
- **Feature Engineering**: Derive domain-specific features (e.g., transaction frequency, location mismatch, transaction time) to enhance model understanding and prediction quality.
- **Threshold Optimization**: Tune the decision threshold to find a better balance between precision and recall, especially in production settings.
- **Real-time and Streaming Detection**: Adapt models to work with real-time transaction streams and implement online learning for continuous fraud adaptation.
- **Explainable AI (XAI)**: Integrate model interpretation tools (e.g., SHAP, LIME) to explain why certain transactions are flagged, improving transparency and trust.

8. Appendix: Source Code

Below is the source code for the project:

```
# Step 1: Import libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

# Step 2: Load your dataset
file_path = '/content/drive/MyDrive/ML assignment data/creditcard.csv'
data = pd.read_csv(file_path)

# Step 3: Prepare features and labels
X = data.drop('Class', axis=1)  # 'Class' column is the target (0 = normal, 1 = fraud)
y = data['Class']

# Step 4: Train/test split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
stratify=y)


# Step 5: Feature scaling

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)


# Step 6: Train Logistic Regression model

model = LogisticRegression(class_weight='balanced', random_state=42)  # class_weight
helps with imbalanced data

model.fit(X_train_scaled, y_train)


# Step 7: Predictions

y_pred = model.predict(X_test_scaled)


# Step 8: Evaluation

print(confusion_matrix(y_test, y_pred))

print(classification_report(y_test, y_pred))

# Step 9: Plot Confusion Matrix

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import ConfusionMatrixDisplay


# Generate confusion matrix

cm = confusion_matrix(y_test, y_pred)


# Plot using seaborn

plt.figure(figsize=(6,4))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
```

```
plt.title('Confusion Matrix - Logistic Regression')

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.show()
```