

▼ Problem: Credit Card Fraud Detection

Credit card fraud is a critical issue faced by financial institutions and consumers worldwide. Fraudulent transactions cause significant financial losses and help reduce trust in payment systems. Detecting fraudulent transactions in real time is challenging due to the high volume of transactions and the evolving tactics used by scammers.

Machine learning provides a powerful approach to detect credit card fraud by using historical transaction data to identify patterns indicative of fraud. Here's a step by step on how we can solve this issue using machine learning:

1. Problem Identification

The goal of this project is to develop a **binary classification model** that is able to predict whether a transaction is **fraudulent** or **legitimate**

2. Data Collection and Preparation

- **Dataset Source:** [Kaggle Credit Card Fraud Dataset](#)
- **Dataset Description:**
 - Contains over **280,000 transactions**.
 - Includes **features** such as `Amount`, `Time`, and 28 anonymized numerical features (`v1` to `v28`).
 - A binary target variable (`Class`), where:
 - `0` = Legitimate Transaction
 - `1` = Fraudulent Transaction

3. Model Design and Architecture

- **Model Type:** Neural Network
- **Architecture:**
 - Input layer: 30 features (including `Amount`, `Time`, and the 28 anonymized features).
 - Hidden layers: 2 configured fully connected hidden layer and ReLU activation.
 - Output layer: A single neuron with **Sigmoid activation** for binary classification.
- **Loss Function:** Binary Cross-Entropy Loss.

4. Training and Testing

Steps:

1. Split the dataset into **train**, and **test** sets.
2. Train the model using **train** data.
3. Test the model using **test** data.
4. Deploy the trained model for real-time predictions.

```
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
import pandas as pd
from google.colab import drive
import pytorch_lightning as pl
from torchmetrics.classification import Accuracy, Precision, Recall
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from torch.utils.data import DataLoader, TensorDataset
from sklearn.preprocessing import StandardScaler
import joblib
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
```

▼ Loading and Initial Exploration of the Credit Card Fraud Dataset

```
drive.mount('/gdrive')
path = '/gdrive/My Drive/creditcard.csv'
df = pd.read_csv(path)

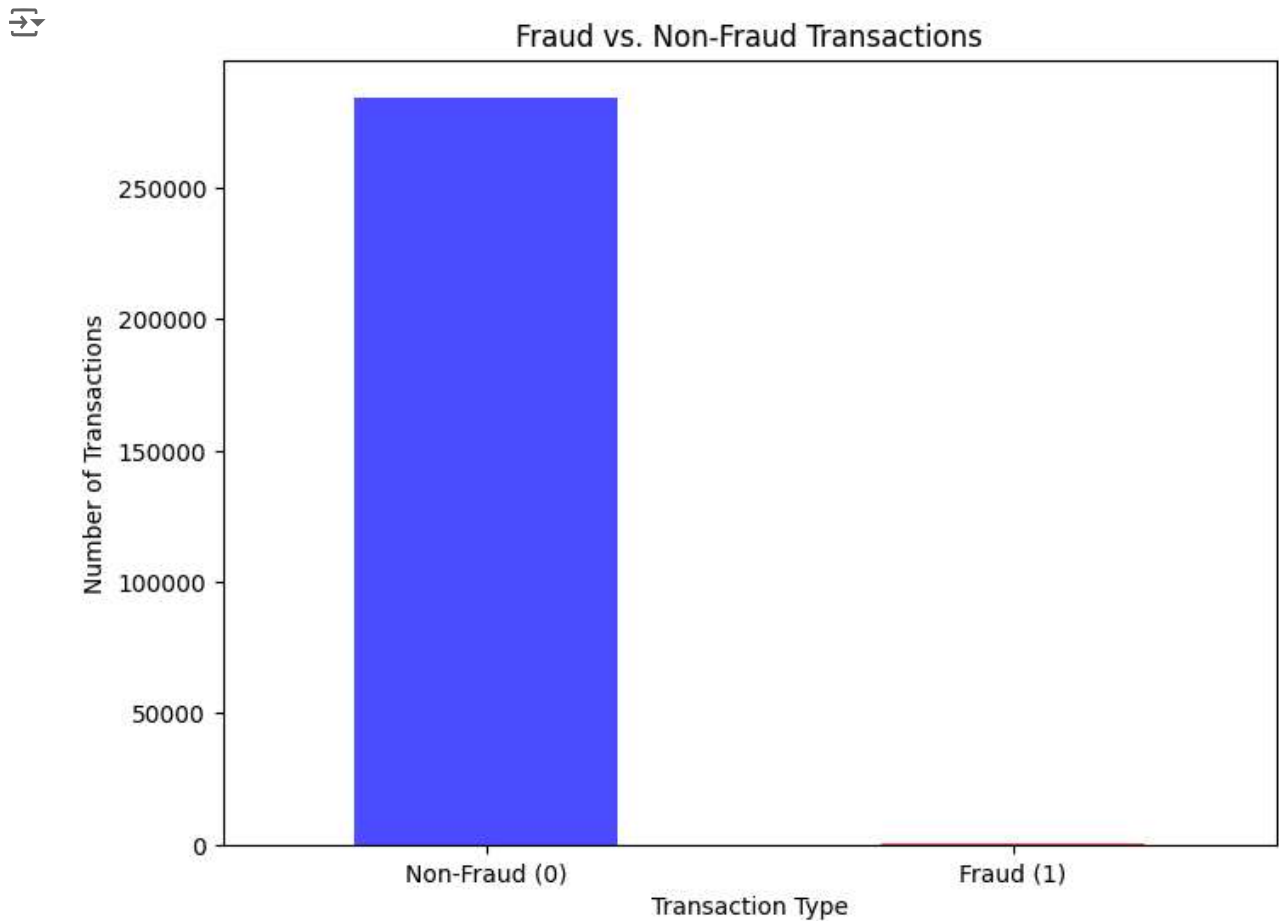
# print( len(df) /(len(df) +len(df[df['Class'] == 1])) )
df[['Time', 'V1', 'V2', 'V3', 'V27', 'V28', 'Amount', 'Class']].head()
```

📁 Drive already mounted at /gdrive; to attempt to forcibly remount, call drive.mount("/gdrive", force_remount=True).

	Time	V1	V2	V3	V27	V28	Amount	Class	📊
0	0.0	-1.359807	-0.072781	2.536347	0.133558	-0.021053	149.62	0	📈
1	0.0	1.191857	0.266151	0.166480	-0.008983	0.014724	2.69	0	
2	1.0	-1.358354	-1.340163	1.773209	-0.055353	-0.059752	378.66	0	
3	1.0	-0.966272	-0.185226	1.792993	0.062723	0.061458	123.50	0	
4	2.0	-1.158233	0.877737	1.548718	0.219422	0.215153	69.99	0	

```
fraud_counts = df['Class'].value_counts()
```

```
plt.figure(figsize=(8, 6))
fraud_counts.plot(kind='bar', color=['blue', 'red'], alpha=0.7)
plt.xticks(ticks=[0, 1], labels=['Non-Fraud (0)', 'Fraud (1)'], rotation=0)
plt.xlabel('Transaction Type')
plt.ylabel('Number of Transactions')
plt.title('Fraud vs. Non-Fraud Transactions')
plt.show()
```



Dataset Imbalance

The dataset exhibits an extremelyclass imbalance. There are over 280,000 non-fraudulent transactions, while fraudulent transactions are less than 500. Initially Model training was impacted by this issue, which led to all its predictions being classified as non-fraudulent and receiving a precision of 0.00 percent.

To address this issue, we used a scaler object from the scikit-learn library to scale the data effectively.

Feature and Target Variable Extraction

```
x = df.drop('Class', axis=1)
y = df['Class']
```

Data Scaling for Improved Model Performance

```
# Scale the data
scaler = StandardScaler()
x = scaler.fit_transform(x)
joblib.dump(scaler, "scaler.pkl")
```

```
['scaler.pkl']
```

Data Splitting and Tensor Conversion


```
y = y.values
```

```
# Split data into train/test sets
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42,stratify=y)
print(f' Number of tests {len(y_test)}')
```

```
# Convert data into PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test, dtype=torch.float32)
```

```
# Create DataLoaders
train_dataset = TensorDataset(X_train_tensor, y_train_tensor)
val_dataset = TensorDataset(X_test_tensor, y_test_tensor)
```

```
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=32)
```

 Number of tests 56962

Model Architecture

```
class FraudDetectionModel(pl.LightningModule):
    def __init__(self, input_size=30, hidden_sizes=[94, 42], lr=1e-3):
        super(FraudDetectionModel, self).__init__()
        self.lr = lr

        self.model = nn.Sequential(
            nn.Linear(input_size, hidden_sizes[0]),
            nn.ReLU(),
            nn.Linear(hidden_sizes[0], hidden_sizes[1]),
            nn.ReLU(),
            nn.Linear(hidden_sizes[1], 1),
            nn.Sigmoid()
        )

        self.accuracy = Accuracy(task="binary")
        self.precision = Precision(task="binary")
        self.recall = Recall(task="binary")

    def forward(self, x):
        return self.model(x)

    def training_step(self, batch, batch_idx):
        x, y = batch
        y_pred = self(x).squeeze() # Remove extra dimension
        loss = F.binary_cross_entropy(y_pred, y)
        self.log("train_loss", loss, on_epoch=True)
        return loss

    def validation_step(self, batch, batch_idx):
        x, y = batch
        y_pred = self(x).squeeze()
        loss = F.binary_cross_entropy(y_pred, y)
        self.log("val_loss", loss, on_epoch=True)
        self.log("val_acc", self.accuracy(y_pred, y), on_epoch=True)
        self.log("val_precision", self.precision(y_pred, y), on_epoch=True)
        self.log("val_recall", self.recall(y_pred, y), on_epoch=True)


    def configure_optimizers(self):
        return torch.optim.Adam(self.parameters(), lr=self.lr)
```

Model Training and Validation

```
model = FraudDetectionModel(input_size=X_train.shape[1], lr=1e-3)
```

```
# Set up the trainer
trainer = pl.Trainer(max_epochs=6)
```


```
# Train the model
trainer.fit(model, train_loader, val_loader)
print(f"Final Accuracy: {trainer.callback_metrics['val_acc']:.4f}")
print(f"Final Precision: {trainer.callback_metrics['val_precision']:.4f}")
print(f"Final Recall: {trainer.callback_metrics['val_recall']:.4f}")
print(f"Final Loss: {trainer.callback_metrics['val_loss']:.4f}")
```




```
INFO:pytorch_lightning.utilities.rank_zero:GPU available: True (cuda), used: True
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False, using: 0 HPUs
INFO:pytorch_lightning.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:pytorch_lightning.callbacks.model_summary:
  | Name      | Type                | Params | Mode
-----|-----|-----|-----|-----
0 | model      | Sequential          | 6.9 K  | train
1 | accuracy   | BinaryAccuracy      | 0      | train
2 | precision  | BinaryPrecision     | 0      | train
3 | recall     | BinaryRecall        | 0      | train
-----|-----|-----|-----|-----
6.9 K   Trainable params
0       Non-trainable params
6.9 K   Total params
0.028   Total estimated model params size (MB)
10      Modules in train mode
0       Modules in eval mode

Epoch 5: 100%
7121/7121 [00:44<00:00, 159.94it/s, v_num=1]

INFO:pytorch_lightning.utilities.rank_zero:`Trainer.fit` stopped: `max_epochs=6` reached.
Final Accuracy: 0.9994
Final Precision: 0.0416
Final Recall: 0.0416
```





Model Testing

```
# Load the test data into DataLoader
test_dataset = TensorDataset(X_test_tensor, y_test_tensor)
test_loader = DataLoader(test_dataset, batch_size=32)
print(y_test_tensor.values)

model.eval() # Set the model to evaluation mode

all_preds = []
all_labels = []

with torch.no_grad():
    for batch in test_loader:
        x, y = batch
        y_hat = model(x).squeeze() # Get model predictions
        all_preds.append(y_hat)
        all_labels.append(y)

all_preds = torch.cat(all_preds).cpu()
all_labels = torch.cat(all_labels).cpu()

all_preds_binary = (all_preds > 0.5).float()

accuracy = accuracy_score(all_labels, all_preds_binary)
precision = precision_score(all_labels, all_preds_binary)

# Compute confusion matrix
cm = confusion_matrix(all_labels, all_preds_binary)

# cm is a 2x2 matrix:
# [ [TN, FP],
#   [FN, TP] ]
# Extract False Negatives from the confusion matrix
```

<built-in method values of Tensor object at 0x7ff4daee5260>

Performance Metrics and Analysis

```
print(f"\nConfusion matrix")
print(f"True Negatives: {cm[0, 0]}")
print(f"False Positives: {cm[0, 1]}")
print(f"False Negatives: {cm[1, 0]}")
print(f"True Positives: {cm[1, 1]}")

print("\nFinal Metrics")
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
```

Confusion matrix
True Negatives: 56854
False Positives: 10
False Negatives: 22
True Positives: 76

Final Metrics
Accuracy: 0.9994
Precision: 0.8837

Saving Model

```
# Save the model's state_dict
torch.save(model.state_dict(), "fraud_detection_model.pth")
from google.colab import files
files.download("fraud_detection_model.pth")
files.download("scaler.pkl")
```

```
! pip install torch pytorch-lightning
```

Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.5.1+cu121)
Collecting pytorch-lightning
 Downloading pytorch_lightning-2.4.0-py3-none-any.whl.metadata (21 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch) (4.12.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.4.2)
Requirement already satisfied: Jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2024.10.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-packages (from torch) (1.13.1)

12/12/24, 3:33 PM

Final_project (1).ipynb - Colab

Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy==1.13.1->torch) (1.3.0)
Requirement already satisfied: tqdm>=4.57.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning) (4.66.6)
Requirement already satisfied: PyYAML>=5.4 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning) (6.0.2)
Collecting torchmetrics>=0.7.0 (from pytorch-lightning)
 Downloading torchmetrics-1.6.0-py3-none-any.whl.metadata (20 kB)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning) (24.2)
Collecting lightning-utilities>=0.10.0 (from pytorch-lightning)
 Downloading lightning_utilities-0.11.9-py3-none-any.whl.metadata (5.2 kB)
Requirement already satisfied: aiohttp!=4.0.0a0,!4.0.0a1 in /usr/local/lib/python3.10/dist-packages (from fsspec[http]>=2022.5.0->pytorch-lig
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from lightning-utilities>=0.10.0->pytorch-lightning) (75
Requirement already satisfied: numpy>1.20.0 in /usr/local/lib/python3.10/dist-packages (from torchmetrics>=0.7.0->pytorch-lightning) (1.26.4)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch) (3.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!4.0.0a1->fsspec[htt
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!4.0.0a1->fsspec[http]>=202
Requirement already satisfied: async-timeout<6.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!4.0.0a1->fsspec[htt
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!4.0.0a1->fsspec[http]>=2022.5
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!4.0.0a1->fsspec[http]>=20
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!4.0.0a1->fsspec[http]>=20
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!4.0.0a1->fsspec[http]>=202
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!4.0.0a1->fsspec[http]>=20
Requirement already satisfied: idna>=2.0 in /usr/local/lib/python3.10/dist-packages (from yarl<2.0,>=1.17.0->aiohttp!=4.0.0a0,!4.0.0a1->fsspe
Downloading pytorch_lightning-2.4.0-py3-none-any.whl (815 kB)
 815.2/815.2 kB 24.8 MB/s eta 0:00:00
Downloading lightning_utilities-0.11.9-py3-none-any.whl (28 kB)
Downloading torchmetrics-1.6.0-py3-none-any.whl (926 kB)
 926.4/926.4 kB 47.0 MB/s eta 0:00:00
Installing collected packages: lightning-utilities, torchmetrics, pytorch-lightning
Successfully installed lightning-utilities-0.11.9 pytorch-lightning-2.4.0 torchmetrics-1.6.0

https://colab.research.google.com/drive/1jG_Dg7S8oYrImTz3ua5dzh13L9L6nKSb#scrollTo=iEMPP2aTD6iN

5/5