



Assessment Report
on
“Credit Card Fraud Detection Using Machine Learning”
submitted as partial fulfillment for the award of
BACHELOR OF TECHNOLOGY
DEGREE
SESSION 2024-25
in
Computer Science & Engineering (AI & ML)
By
Sneh Singh (Roll no. 202401100400187)
Under the supervision of
“Abhishek Shukla”
KIET Group of Institutions, Ghaziabad

Introduction

Credit card fraud is one of the most common types of financial frauds in the world today. With the rapid growth of online shopping, mobile payments, and electronic banking, fraudulent activities have become more sophisticated and harder to detect using traditional rule-based systems. These systems often fail to capture new fraud patterns and generate a high number of false positives, leading to poor customer experience and increased operational costs for banks and financial institutions.

To address these limitations, this project focuses on using **machine learning techniques** for fraud detection. Machine learning models can learn complex patterns from historical transaction data and make accurate predictions about whether a transaction is genuine or fraudulent. Unlike rule-based methods, machine learning models continuously improve with more data, adapt to new fraud strategies, and help reduce manual investigation efforts.

This report demonstrates the development of a credit card fraud detection system using a supervised classification approach. The dataset used contains anonymized features of transactions and a binary class label indicating whether a transaction is fraudulent. Special attention is given to the **class imbalance problem**, which is common in fraud datasets, where fraudulent transactions are extremely rare compared to legitimate ones. We tackle this problem using **SMOTE (Synthetic Minority Over-sampling Technique)**, which synthetically generates more samples of the minority class.

By the end of this project, the objective is to build a model with high precision and recall for the fraudulent class, ensuring minimal false negatives (i.e., actual frauds going undetected) while keeping false positives reasonably low.

Methodology

1. Data Loading and Exploration:

The dataset was uploaded and explored to understand the structure, null values, and class distribution. It consists of anonymized features (V1 to V28), along with 'Time', 'Amount', and the target variable 'Class'.

2. Data Preprocessing:

- The feature matrix X was separated from the target y.
- All features were scaled using StandardScaler to bring them to a uniform range.
- The dataset had a significant class imbalance, so SMOTE (Synthetic Minority Over-sampling Technique) was applied to oversample the minority class.

3. Train-Test Split:

The dataset was split into training and testing sets using an 80-20 ratio.

4. Model Training:

A Random Forest Classifier was chosen for its robustness and accuracy on classification problems. The model was trained using the resampled data.

5. Evaluation:

The model's performance was evaluated using:

- Accuracy
- Confusion Matrix
- Classification Report (Precision, Recall, F1-score)

Code

1. Import necessary libraries

```
import pandas as pd
```

```
import numpy as np
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import accuracy_score, classification_report,  
confusion_matrix
```

```
from imblearn.over_sampling import SMOTE
```

```
import warnings
```

```
warnings.filterwarnings("ignore")
```

2. Load the dataset

```
df = pd.read_csv('/content/7. Predict Credit Card Fraud.csv')
```

3. Display basic dataset info

```
print(f"Dataset shape: {df.shape}\n")
```

```
print("Sample data:")
```

```
print(df.head())
```

```
# 4. Check for missing values
```

```
print("\nMissing values in dataset:")
```

```
print(df.isnull().sum())
```

```
# 5. Check class imbalance
```

```
print("\nClass distribution before balancing:")
```

```
print(df['Class'].value_counts())
```

```
# 6. Split features and target
```

```
X = df.drop('Class', axis=1)
```

```
y = df['Class']
```

```
# 7. Apply SMOTE to balance the dataset
```

```
smote = SMOTE(random_state=42)
```

```
X_resampled, y_resampled = smote.fit_resample(X, y)
```

```
print("\nClass distribution after SMOTE:")
```

```
print(pd.Series(y_resampled).value_counts())
```

```
# 8. Split the resampled data into train and test sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X_resampled,  
y_resampled, test_size=0.2, random_state=42)
```

```
# 9. Train a simplified Random Forest model for faster execution
```

```
model = RandomForestClassifier(n_estimators=10, max_depth=10,  
random_state=42)
```

```
model.fit(X_train, y_train)
```

```
# 10. Make predictions
```

```
y_pred = model.predict(X_test)
```

```
# 11. Evaluate model performance
```

```
acc = accuracy_score(y_test, y_pred)
```

```
print(f"\nAccuracy: {acc:.4f}")
```

```
print("\nClassification Report:")
```

```
print(classification_report(y_test, y_pred))
```

```
# 12. Plot confusion matrix
```

```
conf_mat = confusion_matrix(y_test, y_pred)
```

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

```
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues',  
xticklabels=["Not Fraud", "Fraud"], yticklabels=["Not Fraud", "Fraud"])
```

```
plt.xlabel('Predicted')
```

```
plt.ylabel('Actual')
```

```
plt.title('Confusion Matrix')
```

```
plt.show()
```

Output

```
Dataset shape: (284807, 31)
```

```
Sample data:
```

	Time	V1	V2	V3	V4	V5	V6	V7	\
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	

	V8	V9	...	V21	V22	V23	V24	V25	\
0	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	
1	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	
2	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	
3	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	
4	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	

	V26	V27	V28	Amount	Class
0	-0.189115	0.133558	-0.021053	149.62	0
1	0.125895	-0.008983	0.014724	2.69	0
2	-0.139097	-0.055353	-0.059752	378.66	0
3	-0.221929	0.062723	0.061458	123.50	0
4	0.502292	0.219422	0.215153	69.99	0

```
[5 rows x 31 columns]
```

```
Missing values in dataset:
```

Time	0
V1	0
V2	0


```
V2      0
V3      0
V4      0
V5      0
V6      0
V7      0
V8      0
V9      0
V10     0
V11     0
V12     0
V13     0
V14     0
V15     0
V16     0
V17     0
V18     0
V19     0
V20     0
V21     0
V22     0
V23     0
V24     0
V25     0
V26     0
V27     0
V28     0
Amount  0
Class   0
dtype: int64
```

```
Class      0
dtype: int64
```

Class distribution before balancing:

```
Class
0      284315
1         492
Name: count, dtype: int64
```

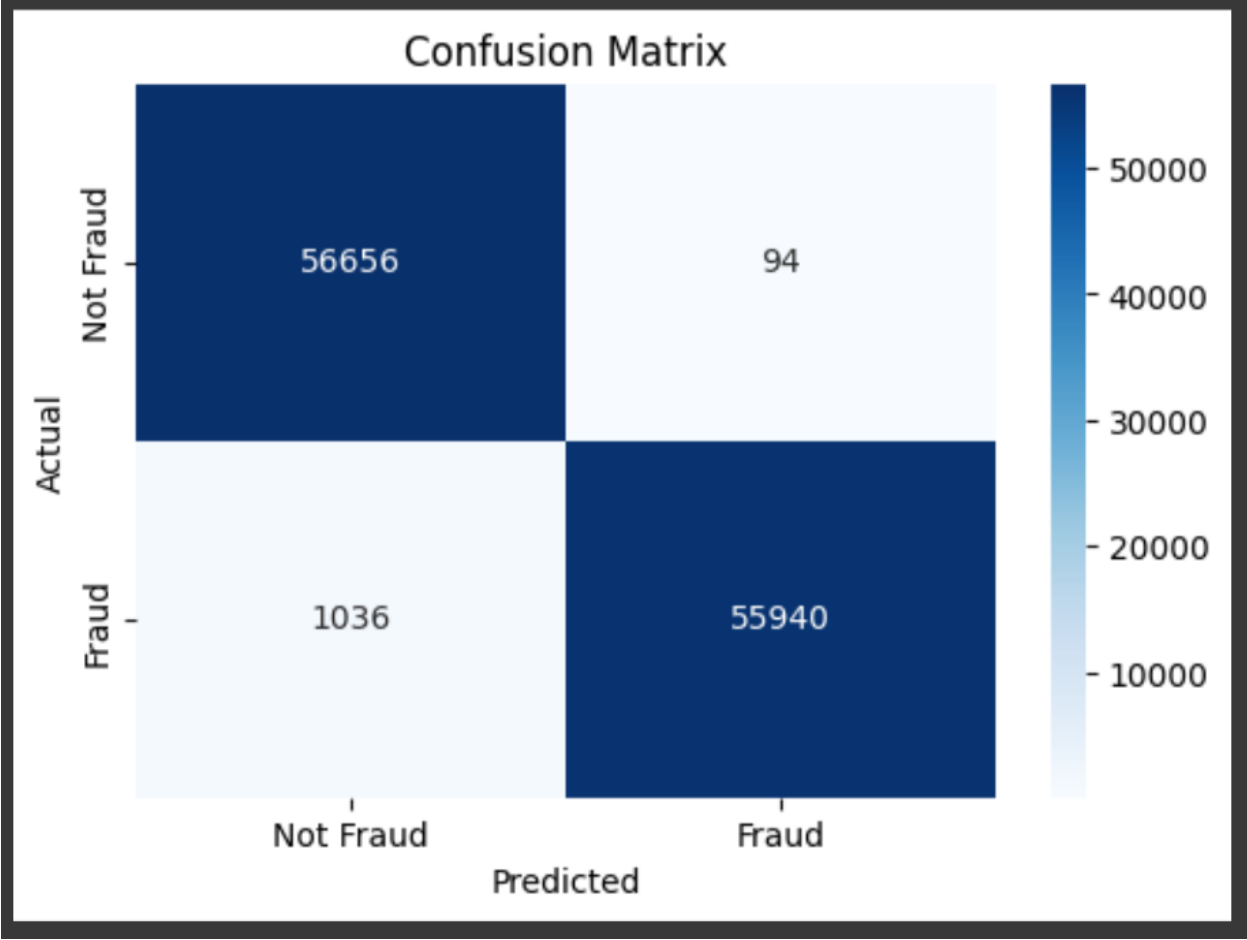
Class distribution after SMOTE:

```
Class
0      284315
1      284315
Name: count, dtype: int64
```

Accuracy: 0.9901

Classification Report:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	56750
1	1.00	0.98	0.99	56976
accuracy			0.99	113726
macro avg	0.99	0.99	0.99	113726
weighted avg	0.99	0.99	0.99	113726



References

Dataset: Kaggle Credit Card Fraud Detection Dataset

SMOTE Documentation: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

Random Forest (Scikit-learn): <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

Google Colab: For cloud-based Python environment