Assesment Report

on

“Problem Statement”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE AIML

By

Naitik Kukreja(2024011004000120)

Under the supervision of

“MR. ABHISHEK SHUKLA”

KIET Group of Institutions, Ghaziabad

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

(Formerly UPTU)

18, May, 2025

Credit Card Fraud Detection

Problem Statement

Credit card fraud is a major issue in the banking and financial sector. Fraudsters use stolen card information to perform unauthorized transactions, leading to financial losses and customer dissatisfaction. The objective of this project is to build a machine learning model that can detect fraudulent credit card transactions based on transaction data.

Dataset Description

The dataset used in this project is sourced from Kaggle and contains credit card transactions made by European cardholders in September 2013. It includes a total of 284,807 transactions, out of which only 492 are fraudulent. This makes the dataset highly imbalanced, with fraud cases representing only 0.17% of the data.

The features in the dataset are mostly anonymized using Principal Component Analysis (PCA), labeled as `V1` to `V28`. Two additional features are:

- `Amount`: The transaction amount.
- `Time`: The time elapsed between this transaction and the first transaction in the dataset.
- `Class`: The target variable, where 0 indicates a legitimate transaction and 1 indicates a fraudulent transaction.

Challenge: Imbalanced Data

Due to the highly imbalanced nature of the dataset, standard classification metrics like accuracy are not sufficient to evaluate model performance. For example, a model that always predicts "not fraud" will still achieve over 99% accuracy. Therefore, metrics like **precision** and **recall** are more informative:

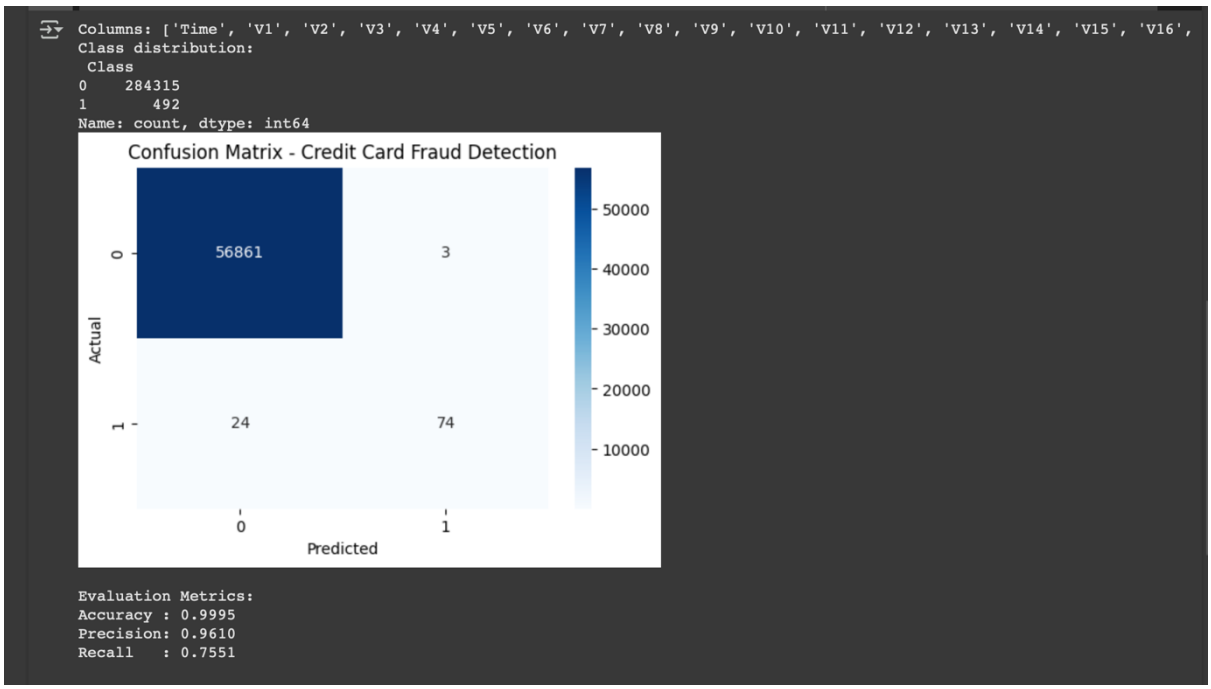
- **Precision**: The proportion of predicted fraudulent transactions that are actually fraud.
- **Recall**: The proportion of actual fraud cases that were correctly identified by the model.

Methodology

1. **Data Preprocessing**
   - Unnecessary columns such as `Time` were dropped.
   - The dataset was split into training and testing sets using stratified sampling to maintain class balance.
2. **Model Training**
   - A Random Forest classifier was used with class balancing to handle the imbalanced data.
3. **Evaluation**
   - A confusion matrix was generated to visualize the model's performance.
   - The model was evaluated using accuracy, precision, and recall metrics.

Conclusion

The credit card fraud detection problem highlights the difficulty of working with imbalanced datasets. While machine learning models such as Random Forest can detect a significant number of fraud cases, careful tuning and evaluation using appropriate metrics are necessary to avoid false positives and minimize missed frauds. This model can assist banks and financial institutions in flagging potentially fraudulent transactions more effectively.



CODE:-

```
# Import necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Import scikit-learn modules for modeling and evaluation
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score

# 1. Load the dataset with appropriate encoding
# The dataset may contain special characters, so we use 'ISO-8859-1' to avoid decoding errors
df = pd.read_csv('/content/7. Predict Credit Card Fraud.csv', encoding='ISO-8859-1')  # treating as credit_fraud.csv

# 2. Preview column names and check class distribution to understand data imbalance
print("Columns:", df.columns.tolist())
print("Class distribution:\n", df['Class'].value_counts())  # Shows how many fraud (1) and non-fraud (0) samples

# 3. Define feature matrix X and target variable y
# Drop the 'Class' column from features (since it's the label), and optionally drop 'Time' if present
X = df.drop(['Class', 'Time'], axis=1, errors='ignore')  # 'errors=ignore' prevents crash if 'Time' not in dataset
y = df['Class']  # Target label: 0 = Not Fraud, 1 = Fraud

# 4. Split the dataset into training and testing sets (80% train, 20% test)
# Stratify ensures class balance in both sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# 5. Train a Random Forest Classifier
# We use class_weight='balanced' to handle class imbalance
model = RandomForestClassifier(n_estimators=100, class_weight='balanced', random_state=42)
model.fit(X_train, y_train)  # Fit model on training data
```

```
# 6. Make predictions on the test set
y_pred = model.predict(X_test)

# 7. Compute the confusion matrix to evaluate predictions
cm = confusion_matrix(y_test, y_pred)

# Visualize the confusion matrix as a heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')       # X-axis label
plt.ylabel('Actual')          # Y-axis label
plt.title('Confusion Matrix - Credit Card Fraud Detection')
plt.show()

# 8. Compute and display evaluation metrics
accuracy = accuracy_score(y_test, y_pred)       # Overall correctness
precision = precision_score(y_test, y_pred)     # Correctly predicted frauds out of total
predicted frauds
recall = recall_score(y_test, y_pred)           # Correctly predicted frauds out of actual frauds

# Display the metrics
print("\nEvaluation Metrics:")
print(f"Accuracy : {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall   : {recall:.4f}")
```