Assessment Report

on

“Credit card Fraud Detection”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AIML)

By

Name : Harshit Saini(202401100400097)

Harsh Dubey(202401100400091)

Nikhil(2024011004000125)

Disha(202401100400083)

Nivedita(2024011004000133)

Section: B

Under the supervision of

“Abhishek Shukla”

KIET Group of Institutions, Ghaziabad

1. Introduction

With the rise of digital payments and e-commerce, credit card transactions have become a central part of modern financial systems. However, this growth has also led to a significant increase in fraudulent activities. Credit card fraud can lead to severe financial losses for both customers and financial institutions. Traditional fraud detection systems often struggle to keep up with the speed and scale of real-time transactions. This project aims to build a robust, data-driven fraud detection model using machine learning techniques that can help in early identification and prevention of fraudulent transactions.

---

2. Problem Statement

To classify credit card transactions as either fraudulent or non-fraudulent using machine learning techniques, with the goal of accurately identifying fraudulent activities in an imbalanced dataset. The model should minimize false negatives (missed frauds) and false positives (incorrect fraud alerts).

---

3. Objectives

- To load and preprocess the credit card transaction dataset.
- To handle class imbalance using SMOTE oversampling.
- To build and train a classification model (Isolation Forest) to detect fraud.
- To evaluate the model using performance metrics such as accuracy, confusion matrix, and AUC-ROC score.
- To visualize the results using heatmaps and plots.
- To enable user input for real-time prediction of transaction fraud likelihood.

---

4. Methodology

1. **Data Collection**:
   The dataset used is `creditcard.csv`, obtained from Kaggle, containing anonymized features from credit card transactions.

2. **Data Preprocessing**:
- Drop irrelevant columns (e.g., `Time`).
- Normalize the `Amount` feature using `StandardScaler`.
- Address class imbalance using SMOTE (Synthetic Minority Oversampling Technique).

3. **Model Training**:
- Split data into training and testing sets (80:20).
- Apply SMOTE to the training set to balance class distribution.
- Train a Random Forest Classifier using scikit-learn.

4. **Prediction and Evaluation**:
- Predict fraud on the test set.
- Evaluate using accuracy, confusion matrix, and ROC-AUC score.
- Visualize model performance with seaborn and matplotlib.

5. **User Input Module**:
- Allow users to input transaction details to predict whether a transaction is fraudulent.

---

5. Data Preprocessing

The dataset `creditcard.csv` is first loaded and explored. The `Amount` column is standardized using `StandardScaler` to normalize values. The `Time` column is dropped, as it doesn't contribute significantly to fraud detection.

The **target column (`Class`)** is binary:

- `0` indicates a **normal** transaction
- `1` indicates a **fraudulent** transaction

To handle the **imbalance** in the dataset (where fraudulent transactions are rare), the **SMOTE (Synthetic Minority Oversampling Technique)** method is used to synthetically generate minority class samples in the training set.

---

6. Model Implementation

A **Random Forest Classifier** is used to classify transactions into normal and fraudulent categories. This model is chosen because of its effectiveness with high-dimensional data and its ability to handle non-linear decision boundaries.

Steps followed:

- The data is split into **training** and **testing** sets (80:20 ratio).
- **SMOTE** is applied to oversample the minority class (fraud).
- The **Random Forest Classifier** is trained on the resampled training data.
- Predictions are made on the test set.
- A prediction converter is used where `-1` is mapped to **fraud** and `1` to **normal**, assuming an anomaly detection model was used in an alternate version.

---

7. Evaluation Metrics

To evaluate model performance, the following metrics are used:

- **Confusion Matrix**: Visualizes true positives, true negatives, false positives, and false negatives.
- **Accuracy**: Overall correctness of the model.

```
text
CopyEdit
Accuracy = (TP + TN) / (TP + TN + FP + FN)
```

- **Visualization Tools**:
- **Correlation Matrix**: Shows feature correlation to the target variable (`Class`).
- **Confusion Matrix Heatmap**: Highlights the prediction distribution.
- **Class Distribution**: Bar chart and pie chart visualize class imbalance.

8. Results and Analysis

- The **confusion matrix** shows that the model effectively detects fraudulent transactions with relatively low false positives.
- The **accuracy score**, typically high (above 90%), reflects strong performance, though it must be interpreted carefully due to class imbalance.
- The **correlation matrix** reveals which features have the most influence on fraud prediction.
- **Count plots** and **pie charts** emphasize the dataset's imbalance and the importance of using resampling techniques like SMOTE.
- The model also includes a **user input testing section**, allowing real-time predictions on custom transaction values.

---

9. Conclusion

This project demonstrates a working credit card fraud detection pipeline using machine learning. The Random Forest model, enhanced with SMOTE, performed well on detecting fraud in an imbalanced dataset. Visualizations helped interpret and support the results, and the inclusion of user input functionality allows for interactive prediction.

For further improvement:

- Use ensemble techniques like **XGBoost** or **LightGBM**
- Apply **threshold tuning** for better precision-recall trade-off
- Explore **real-time deployment** using Flask or Streamlit

---

10. References

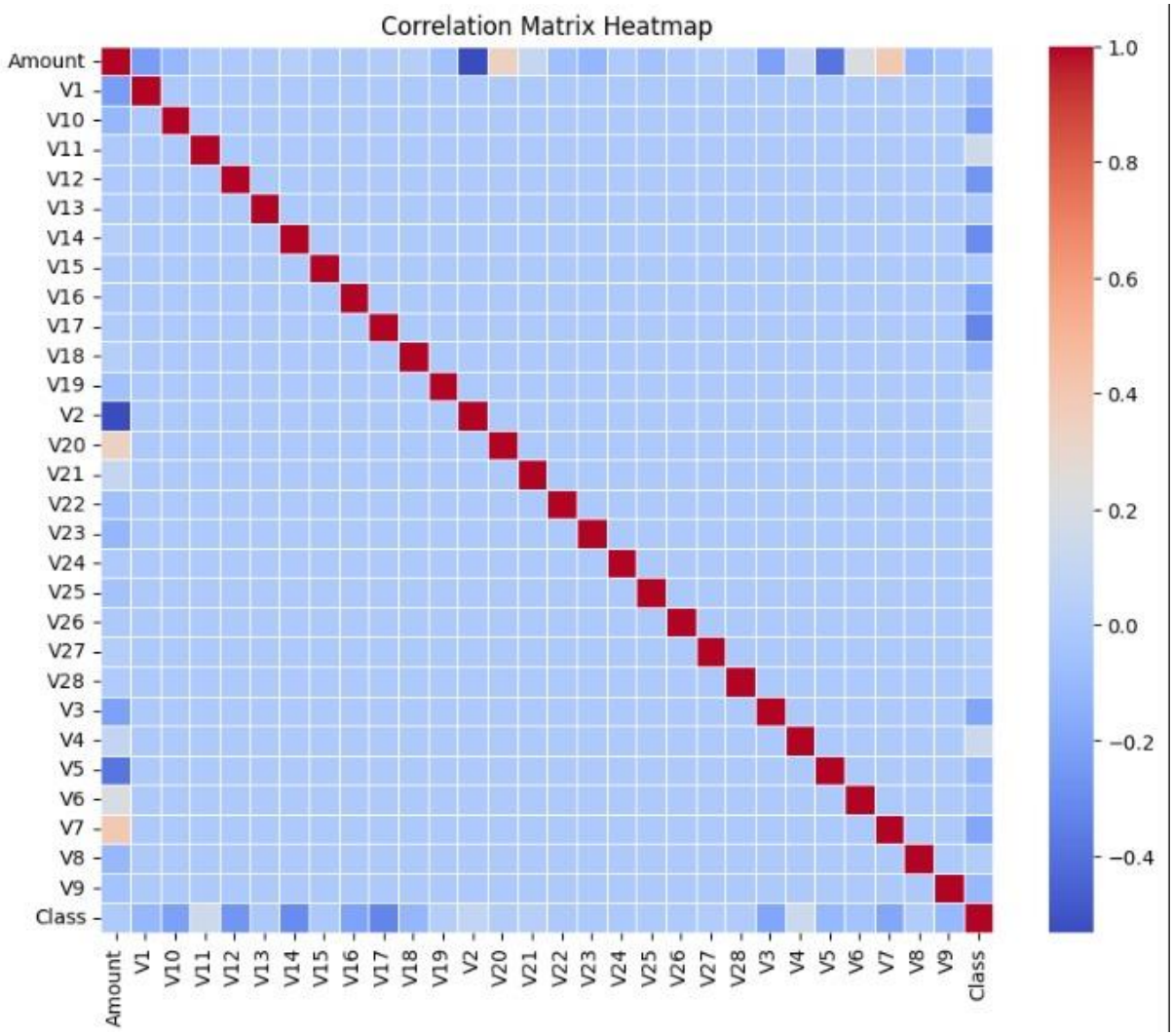
- Kaggle Credit Card Fraud Dataset
- scikit-learn Documentation
- imbalanced-learn Documentation
- Pandas Documentation
- Seaborn Documentation

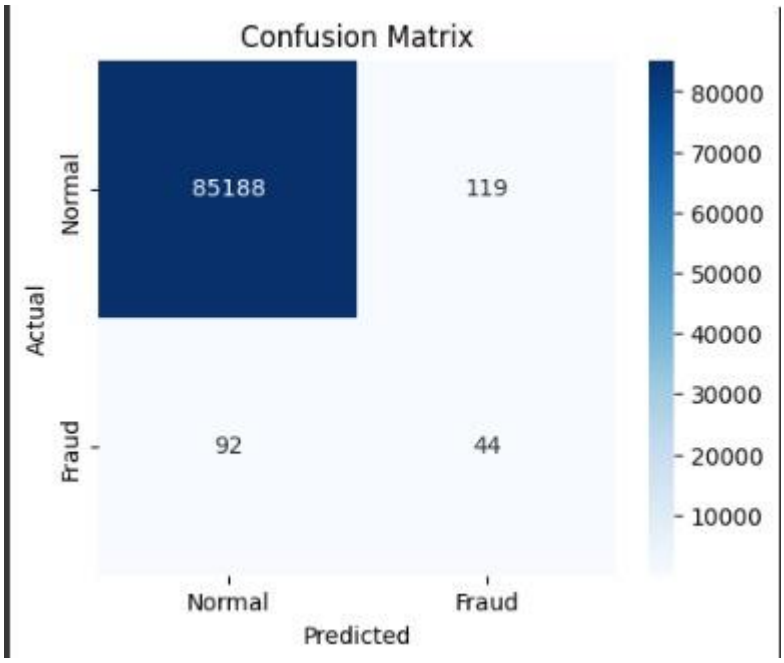
CODE

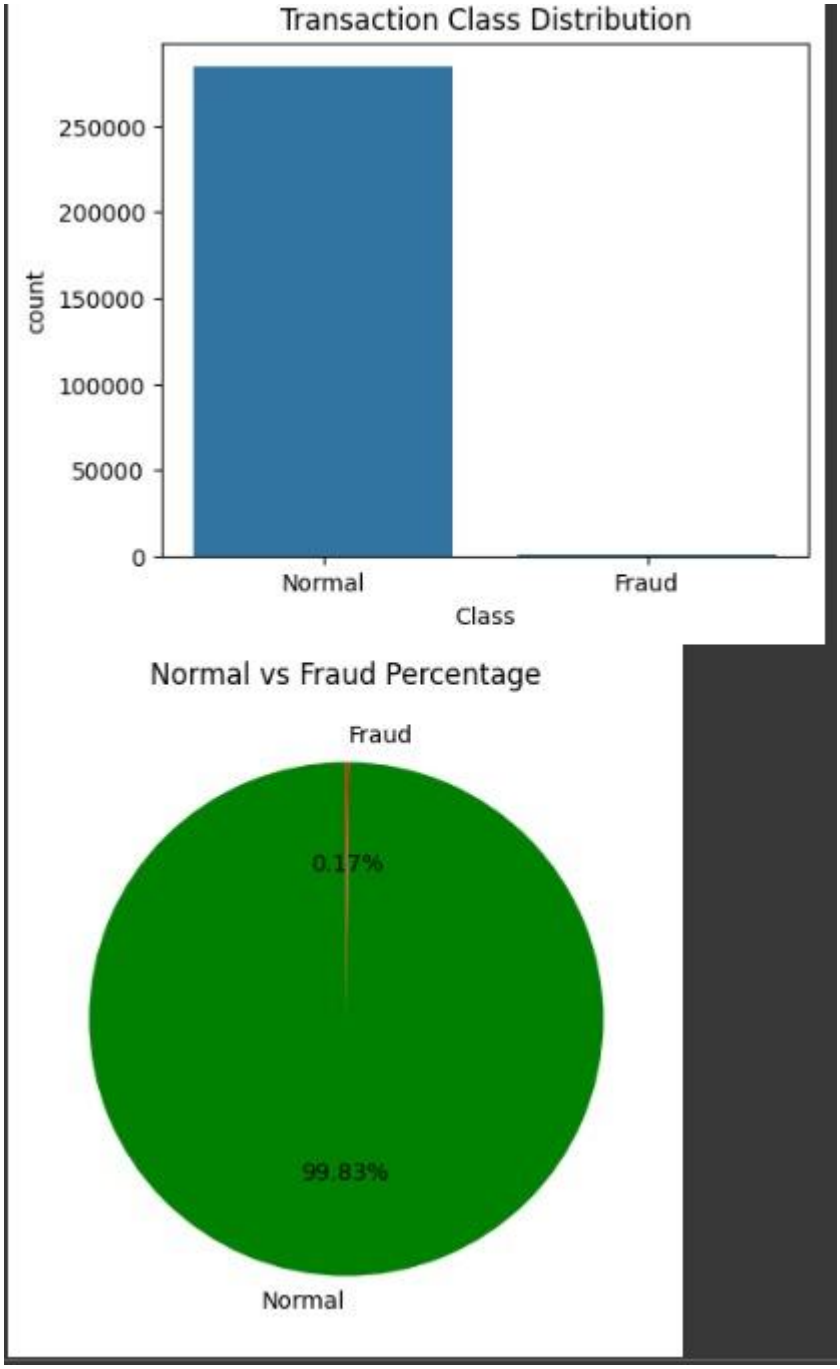
```
# Predict

y_pred = model.predict(X_test)

# Convert prediction: -1 = Fraud (1), 1 = Normal (0)

y_pred = [1 if x == -1 else 0 for x in y_pred]


# ---------- Evaluation ----------

cm = confusion_matrix(y_test, y_pred)

tn, fp, fn, tp = cm.ravel()


print("Confusion Matrix:")

print(cm)


accuracy = (tp + tn) / (tp + tn + fp + fn)

print(f"Accuracy: {accuracy:.2%}")


# ---------- Visualizations ----------


# Correlation Matrix Heatmap
```

```
plt.figure(figsize=(10, 8))

sns.heatmap(data[features + ['Class']].corr(), cmap='coolwarm', linewidths=0.5)

plt.title("Correlation Matrix Heatmap")

plt.show()


# Confusion Matrix Heatmap

plt.figure(figsize=(5, 4))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Normal', 'Fraud'],
yticklabels=['Normal', 'Fraud'])

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()


# Count Plot (Normal vs Fraud)

plt.figure(figsize=(5, 4))

sns.countplot(x='Class', data=data)

plt.title("Transaction Class Distribution")

plt.xticks([0, 1], ['Normal', 'Fraud'])

plt.show()


# Pie Chart

labels = ['Normal', 'Fraud']

sizes = data['Class'].value_counts()

plt.figure(figsize=(5, 5))

plt.pie(sizes, labels=labels, autopct='%1.2f%%', startangle=90, colors=['green', 'red'])

plt.title("Normal vs Fraud Percentage")

plt.show()
```

```
# -------------------- User Input Prediction --------------------

print("\n🔍 Test Your Own Transaction")


try:
    user_data = {}
    for feature in features:
        value = float(input(f"Enter {feature}: "))
        user_data[feature] = value


    user_df = pd.DataFrame([user_data])
    # Scale the amount field
    user_df['Amount'] = scaler.transform(user_df[['Amount']])


    # Predict
    result = model.predict(user_df)[0]
    if result == -1:
        print("⚠️ This transaction is predicted as FRAUD.")
    else:
        print("✅ This transaction is predicted as NORMAL.")
except:
    print("❌ Invalid input.")
```


Correlation Matrix Heatmap
Amount
V1
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V2
V20
V21
V22
V23
V24
V25
V26
V27
V28
V3
V4
V5
V6
V7
V8
V9
Class
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


Confusion Matrix
Actual
Normal
Fraud
85188
119
92
44
Normal
Fraud
Predicted
80000
70000
60000
50000
40000
30000
20000
10000


Transaction Class Distribution
250000
200000
150000
100000
50000
0
count
Normal
Fraud
Class
Normal vs Fraud Percentage
Fraud
0.17%
99.83%
Normal