Documentation: Credit Card Fraud Detection Using Random Forest

---

Overview

This project focuses on detecting fraudulent credit card transactions using a supervised machine learning approach. It uses a real-world dataset and applies a Random Forest Classifier to predict fraudulent transactions. The dataset is highly imbalanced, and various evaluation metrics are used to judge the performance of the model.

---

1. Dataset: `creditcard.csv`

Description:

- Contains transaction data including anonymized features (V1-V28), time, amount, and the class label.
- Class Label:
- 0: Valid Transaction
- 1: Fraudulent Transaction
- Highly imbalanced: Very few fraudulent cases compared to valid ones.

Loading and Processing Data:

```
import pandas as pd

# Load dataset
data = pd.read_csv("creditcard.csv")

# Separate fraud and valid transactions
fraud = data[data['Class'] == 1]
valid = data[data['Class'] == 0]

# Ratio of fraudulent to valid transactions
outlierFraction = len(fraud) / len(valid)
print(outlierFraction)
```

---

2. Feature Preparation

Features and Target:

- Features (X): All columns except 'Class'
- Target (Y): 'Class' column indicating transaction type

```
X = data.drop(['Class'], axis=1)
Y = data['Class']

xData = X.values
yData = Y.values
```

Train/Test Split:

```
from sklearn.model_selection import train_test_split

xTrain, xTest, yTrain, yTest = train_test_split(
    xData, yData, test_size=0.2, random_state=42
)
```

---

3. Model Training: Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

# Initialize and train model
rfc = RandomForestClassifier()
rfc.fit(xTrain, yTrain)

# Predictions
yPred = rfc.predict(xTest)
```

---

4. Evaluation Metrics

Used Metrics:

- Accuracy
- Precision
- Recall
- F1 Score
- Matthews Correlation Coefficient (MCC)

```
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score,
    f1_score, matthews_corrcoef, confusion_matrix
)
```

```
accuracy = accuracy_score(yTest, yPred)
precision = precision_score(yTest, yPred)
recall = recall_score(yTest, yPred)
f1 = f1_score(yTest, yPred)
mcc = matthews_corrcoef(yTest, yPred)

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")
print(f"Matthews Correlation Coefficient: {mcc:.4f}")
```

---

5. Confusion Matrix Visualization

```
import seaborn as sns
import matplotlib.pyplot as plt

conf_matrix = confusion_matrix(yTest, yPred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
            xticklabels=['Normal', 'Fraud'], yticklabels=['Normal', 'Fraud'])
plt.title("Confusion Matrix")
plt.xlabel("Predicted Class")
plt.ylabel("True Class")
plt.show()
```

---

6. Model Saving: Pickle File

The trained model is saved using `joblib` for future use.

```
import joblib

# Save model to a pickle file
joblib.dump(rfc, "fraud_model.pkl")
```

---

7. Prediction Script: `predict_from_csv.py`

Purpose:

- Load the saved model (`fraud_model.pkl`)

- Load a new dataset (CSV format)
- Predict class labels for new transactions

```
import pandas as pd
import joblib

# Load saved model
model = joblib.load("fraud_model.pkl")

# Load new transaction data
new_data = pd.read_csv("new_transactions.csv")

# Predict classes
predictions = model.predict(new_data)

# Print or store predictions
print(predictions)
```

---

8. Supporting Files

- ``: Main dataset for training and testing the model.
- ``: New, unseen data used to test the model after deployment.
- ``: Serialized model file used to load the trained Random Forest model without retraining.

---

9. Use Case & Deployment

This codebase is useful for financial institutions looking to embed fraud detection in their data pipelines. Once trained, the model can be used to scan new transactions in real-time (or batch) and flag fraudulent ones.

By saving the model with `joblib`, you avoid retraining every time and can easily scale predictions. For more automation, you could connect this pipeline to a database or API.

End of Documentation