Credit Card Fraud Detection using Machine Learning

This notebook implements an AI-powered approach to detect credit card fraud. It includes data loading, preprocessing, model training, and evaluation using Logistic Regression, Random Forest, and XGBoost.

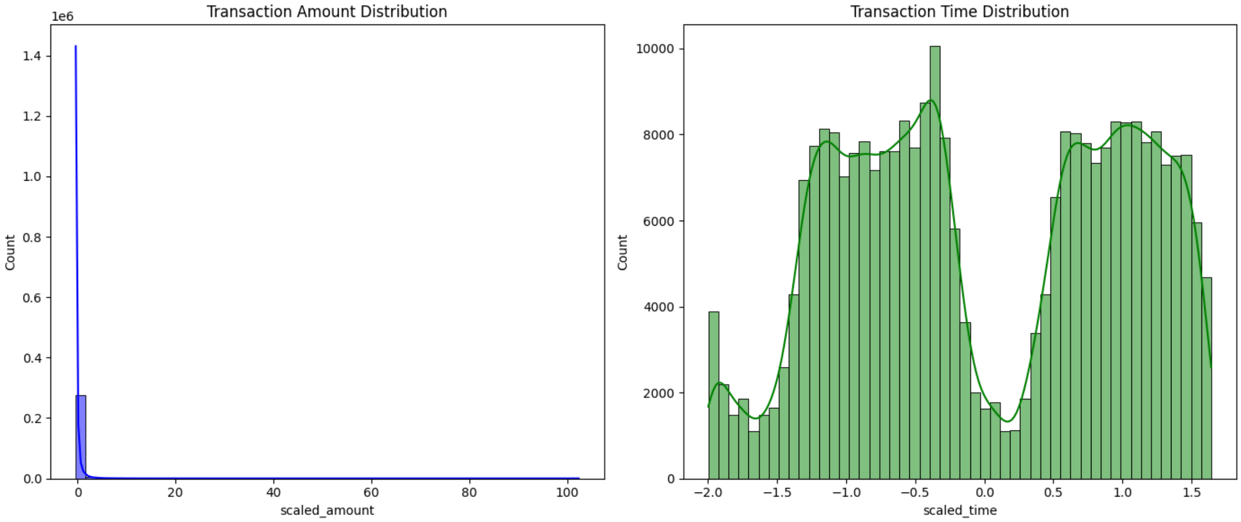
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, roc_auc_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from imblearn.over_sampling import SMOTE
import warnings
warnings.filterwarnings("ignore")

# Load dataset
# Original URL: url = 'https://raw.githubusercontent.com/selva86/datasets/master/creditcard.csv'
# Updated URL:
url = 'https://storage.googleapis.com/download.tensorflow.org/data/creditcard.csv'  # Corrected URL
df = pd.read_csv(url)
df.head()
```

```
{"type":"dataframe","variable_name":"df"}
```

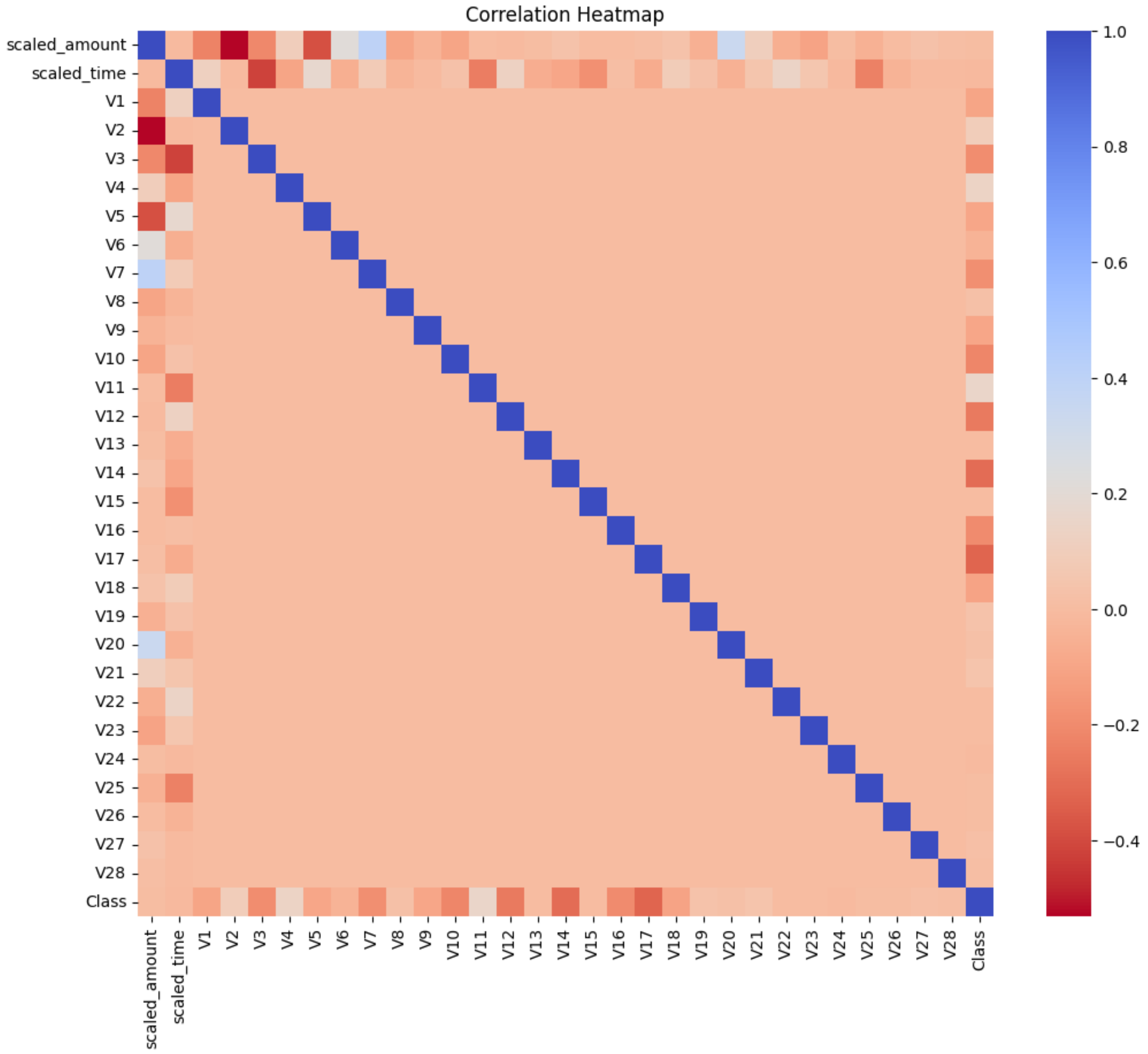
```python
# Feature scaling
scaler = StandardScaler()
df['scaled_amount'] = scaler.fit_transform(df[['Amount']])
df['scaled_time'] = scaler.fit_transform(df[['Time']])
df.drop(['Amount', 'Time'], axis=1, inplace=True)
df = df[['scaled_amount', 'scaled_time'] + [col for col in df.columns if col not in ['scaled_amount', 'scaled_time']]]
df.head()
```

```
{"type":"dataframe","variable_name":"df"}
```

```
# EDA - Histograms
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.histplot(df['scaled_amount'], bins=50, kde=True, color='blue')
plt.title("Transaction Amount Distribution")
plt.subplot(1, 2, 2)
sns.histplot(df['scaled_time'], bins=50, kde=True, color='green')
plt.title("Transaction Time Distribution")
plt.tight_layout()
plt.show()
```



```
# Correlation heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), cmap='coolwarm_r')
plt.title("Correlation Heatmap")
plt.show()
```



```
# Apply SMOTE
X = df.drop('Class', axis=1)
y = df['Class']
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X, y)
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res,
test_size=0.2, stratify=y_res, random_state=42)

# Train and evaluate models
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Random Forest': RandomForestClassifier(),
    'XGBoost': XGBClassifier(use_label_encoder=False,
eval_metric='logloss')
}
```

```
for name, model in models.items():
    print(f"\n{name}")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print("Classification Report:")
    print(classification_report(y_test, y_pred))
    print("ROC AUC Score:", roc_auc_score(y_test,
model.predict_proba(X_test)[:,1]))
```

```
Logistic Regression
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     56863
           1       0.99      0.97      0.98     56863

    accuracy                           0.98    113726
   macro avg       0.98      0.98      0.98    113726
weighted avg       0.98      0.98      0.98    113726

ROC AUC Score: 0.9974108882263335

Random Forest
```