In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import IsolationForest
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score
```

In [3]:

```
df=pd.read_csv("creditcard.csv")
```

In [4]:

```
df
```

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.2
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.7
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.2
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.5
...	...	...	...	...	...	...	...	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.9
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.0
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.2
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.6
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.5

284807 rows × 31 columns

In [5]:

```
df.head()
```

Out[5]:

	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	0.0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.2
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.3
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.2

5 rows × 31 columns

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

In [7]:
```
df.describe()
```

Out[7]:

	Time	V1	V2	V3	V4	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.8
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.6
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.3
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.1
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.9
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.4
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.1
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.4

8 rows × 31 columns

In [8]:

```
df.shape[0]
```

Out[8]:

```
284807
```

In [34]:

```
df.transpose()
```

Out[34]:

	0	1	2	3	4	5	6
Time	0.000000	0.000000	1.000000	1.000000	2.000000	2.000000	4.000000
V1	-1.359807	1.191857	-1.358354	-0.966272	-1.158233	-0.425966	1.229658
V2	-0.072781	0.266151	-1.340163	-0.185226	0.877737	0.960523	0.141004
V3	2.536347	0.166480	1.773209	1.792993	1.548718	1.141109	0.045371
V4	1.378155	0.448154	0.379780	-0.863291	0.403034	-0.168252	1.202613
V5	-0.338321	0.060018	-0.503198	-0.010309	-0.407193	0.420987	0.191881
V6	0.462388	-0.082361	1.800499	1.247203	0.095921	-0.029728	0.272708
V7	0.239599	-0.078803	0.791461	0.237609	0.592941	0.476201	-0.005159
V8	0.098698	0.085102	0.247676	0.377436	-0.270533	0.260314	0.081213
V9	0.363787	-0.255425	-1.514654	-1.387024	0.817739	-0.568671	0.464960
V10	0.090794	-0.166974	0.207643	-0.054952	0.753074	-0.371407	-0.099254
V11	-0.551600	1.612727	0.624501	-0.226487	-0.822843	1.341262	-1.416907
V12	-0.617801	1.065235	0.066084	0.178228	0.538196	0.359894	-0.153826
V13	-0.991390	0.489095	0.717293	0.507757	1.345852	-0.358091	-0.751063
V14	-0.311169	-0.143772	-0.165946	-0.287924	-1.119670	-0.137134	0.167372
V15	1.468177	0.635558	2.345865	-0.631418	0.175121	0.517617	0.050144
V16	-0.470401	0.463917	-2.890083	-1.059647	-0.451449	0.401726	-0.443587
V17	0.207971	-0.114805	1.109969	-0.684093	-0.237033	-0.058133	0.002821
V18	0.025791	-0.183361	-0.121359	1.965775	-0.038195	0.068653	-0.611987
V19	0.403993	-0.145783	-2.261857	-1.232622	0.803487	-0.033194	-0.045575
V20	0.251412	-0.069083	0.524980	-0.208038	0.408542	0.084968	-0.219633
V21	-0.018307	-0.225775	0.247998	-0.108300	-0.009431	-0.208254	-0.167716
V22	0.277838	-0.638672	0.771679	0.005274	0.798278	-0.559825	-0.270710
V23	-0.110474	0.101288	0.909412	-0.190321	-0.137458	-0.026398	-0.154104
V24	0.066928	-0.339846	-0.689281	-1.175575	0.141267	-0.371427	-0.780055
V25	0.128539	0.167170	-0.327642	0.647376	-0.206010	-0.232794	0.750137
V26	-0.189115	0.125895	-0.139097	-0.221929	0.502292	0.105915	-0.257237
V27	0.133558	-0.008983	-0.055353	0.062723	0.219422	0.253844	0.034507
V28	-0.021053	0.014724	-0.059752	0.061458	0.215153	0.081080	0.005168
Amount	149.620000	2.690000	378.660000	123.500000	69.990000	3.670000	4.990000
Class	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
anomaly	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

32 rows × 284807 columns

In [9]:
```
df.shape
```

Out[9]:
```
(284807, 31)
```

In [10]:
```
print(df.isnull().sum())
```

```
Time      0
V1        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
```

In [11]:
```
df_cleaned = df.dropna()
```

In [20]:
```
features = ['V1', 'V2', 'V3']
X = df[features]
y = df['Amount']
```

In [21]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [23]:
```
isolation_forest = IsolationForest(contamination=0.01)
isolation_forest.fit(X_scaled)
outliers = isolation_forest.predict(X_scaled)
df['anomaly'] = outliers
```

In [24]:
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3,
```

In [28]:

```
# Define the same threshold for the test set
y_test = (y_test > threshold_value).astype(int)

# Now fit the Logistic Regression model
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
y_pred_log = log_reg.predict(X_test)

# Calculate accuracy
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_log))
```

```
Logistic Regression Accuracy: 0.9874068092178412
```

In [29]:

```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
y_pred_tree = decision_tree.predict(X_test)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_tree))
```

```
Decision Tree Accuracy: 0.9775054714839132
```

In [30]:

```
print("Classification Report for Logistic Regression:")
print(classification_report(y_test, y_pred_log))

print("Classification Report for Decision Tree:")
print(classification_report(y_test, y_pred_tree))
```

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

           0       0.00      0.00      0.00      1076
           1       0.99      1.00      0.99     84367

    accuracy                           0.99     85443
   macro avg       0.49      0.50      0.50     85443
weighted avg       0.97      0.99      0.98     85443

Classification Report for Decision Tree:
```

```
C:\Users\rushi\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-define
d and being set to 0.0 in labels with no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\rushi\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-define
d and being set to 0.0 in labels with no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\rushi\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-define
d and being set to 0.0 in labels with no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
              precision    recall  f1-score   support

           0       0.14      0.15      0.14      1076
           1       0.99      0.99      0.99     84367

    accuracy                           0.98     85443
   macro avg       0.56      0.57      0.57     85443
weighted avg       0.98      0.98      0.98     85443
```

In [36]:

```python
import time
import numpy as np

def simulate_real_time_monitoring(data, model, scaler, threshold=0.5):
    scaled_data = scaler.transform(data)
    fraud_probs = model.predict_proba(scaled_data)[:, 1]
    fraud_preds = (fraud_probs > threshold).astype(int)

    for i, pred in enumerate(fraud_preds):
        print(f"Transaction {i}: {'Fraudulent' if pred else 'Normal'}")

    print("-" * 40)
    time.sleep(2)

for batch_num in range(3):
    print(f"Processing Batch {batch_num + 1}...")
    new_data = np.random.rand(10, 3)
    simulate_real_time_monitoring(new_data, log_reg, scaler)
    print("\n")
```

```
Processing Batch 1...
Transaction 0: Fraudulent
Transaction 1: Fraudulent
Transaction 2: Fraudulent
Transaction 3: Fraudulent
Transaction 4: Fraudulent
Transaction 5: Fraudulent
Transaction 6: Fraudulent
Transaction 7: Fraudulent
Transaction 8: Fraudulent
Transaction 9: Fraudulent
----------------------------------------
```

```
C:\Users\rushi\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but StandardScale
r was fitted with feature names
  warnings.warn(
```

```

Processing Batch 2...
Transaction 0: Fraudulent
Transaction 1: Fraudulent
Transaction 2: Fraudulent
Transaction 3: Fraudulent
Transaction 4: Fraudulent
Transaction 5: Fraudulent
Transaction 6: Fraudulent
Transaction 7: Fraudulent
Transaction 8: Fraudulent
Transaction 9: Fraudulent
----------------------------------------
```

```
C:\Users\rushi\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but StandardScale
r was fitted with feature names
  warnings.warn(
```

```
Processing Batch 3...
Transaction 0: Fraudulent
Transaction 1: Fraudulent
Transaction 2: Fraudulent
Transaction 3: Fraudulent
Transaction 4: Fraudulent
Transaction 5: Fraudulent
Transaction 6: Fraudulent
Transaction 7: Fraudulent
Transaction 8: Fraudulent
Transaction 9: Fraudulent
----------------------------------------
```

```
C:\Users\rushi\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but StandardScale
r was fitted with feature names
  warnings.warn(
```

In [35]:

```python
def check_transactions(data, model, scaler, threshold=0.5):
    predictions = (model.predict_proba(scaler.transform(data))[:, 1] > threshold
    for i, pred in enumerate(predictions):
        print(f"Transaction {i}: {'Fraudulent' if pred else 'Normal'}")
    time.sleep(2)  # Simulate delay

# Process 5 batches of 10 transactions
for batch_num in range(5):
    print(f"Checking Batch {batch_num + 1}...")
    check_transactions(np.random.rand(10, 3), log_reg, scaler)
    print("\n")
```

```
Checking Batch 1...
Transaction 0: Fraudulent
Transaction 1: Fraudulent
Transaction 2: Fraudulent
Transaction 3: Fraudulent
Transaction 4: Fraudulent
Transaction 5: Fraudulent
Transaction 6: Fraudulent
Transaction 7: Fraudulent
Transaction 8: Fraudulent
Transaction 9: Fraudulent
```

```
C:\Users\rushi\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but StandardScale
r was fitted with feature names
  warnings.warn(
Checking Batch 2...
Transaction 0: Fraudulent
Transaction 1: Fraudulent
Transaction 2: Fraudulent
Transaction 3: Fraudulent
Transaction 4: Fraudulent
Transaction 5: Fraudulent
Transaction 6: Fraudulent
Transaction 7: Fraudulent
Transaction 8: Fraudulent
Transaction 9: Fraudulent
```

```
C:\Users\rushi\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but StandardScale
r was fitted with feature names
  warnings.warn(
```

```
Checking Batch 3...
Transaction 0: Fraudulent
Transaction 1: Fraudulent
Transaction 2: Fraudulent
Transaction 3: Fraudulent
Transaction 4: Fraudulent
Transaction 5: Fraudulent
Transaction 6: Fraudulent
Transaction 7: Fraudulent
Transaction 8: Fraudulent
Transaction 9: Fraudulent
```

```
C:\Users\rushi\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but StandardScale
r was fitted with feature names
  warnings.warn(
```

```
Checking Batch 4...
Transaction 0: Fraudulent
Transaction 1: Fraudulent
Transaction 2: Fraudulent
Transaction 3: Fraudulent
Transaction 4: Fraudulent
Transaction 5: Fraudulent
Transaction 6: Fraudulent
Transaction 7: Fraudulent
Transaction 8: Fraudulent
Transaction 9: Fraudulent
```

```
C:\Users\rushi\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but StandardScale
r was fitted with feature names
  warnings.warn(
Checking Batch 5...
Transaction 0: Fraudulent
Transaction 1: Fraudulent
Transaction 2: Fraudulent
Transaction 3: Fraudulent
Transaction 4: Fraudulent
Transaction 5: Fraudulent
Transaction 6: Fraudulent
Transaction 7: Fraudulent
Transaction 8: Fraudulent
Transaction 9: Fraudulent
```

```
C:\Users\rushi\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but StandardScale
r was fitted with feature names
  warnings.warn(
```

In [37]:

```python
def check_fraud(transaction, model, scaler, threshold=0.5):
    scaled_transaction = scaler.transform([transaction])
    fraud_prob = model.predict_proba(scaled_transaction)[:, 1]
    return fraud_prob[0] > threshold
transaction_data = np.random.rand(3)
if check_fraud(transaction_data, log_reg, scaler):
    print("Fraudulent transaction detected.")
else:
    print("Transaction is normal.")
```

```
Fraudulent transaction detected.
```

```
C:\Users\rushi\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but StandardScale
r was fitted with feature names
  warnings.warn(
```