In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input direc

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you c
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

In [16]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

data = pd.read_csv('/kaggle/input/fraud-detection-with-machine-learning/data.csv')
data

label_encoder = LabelEncoder()
data['paymentMethod'] = label_encoder.fit_transform(data['paymentMethod'])

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

In [17]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

In [18]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [19]:
```
log_reg = LogisticRegression(random_state=42)
random_forest = RandomForestClassifier(random_state=42)
svm = SVC(random_state=42)
```

In [20]:
```
log_reg.fit(X_train, Y_train)
random_forest.fit(X_train, Y_train)
svm.fit(X_train_scaled, Y_train)
```

Out[20]:
```
SVC
SVC(random_state=42)
```

In [21]:
```
y_pred_log_reg = log_reg.predict(X_test)
y_pred_random_forest = random_forest.predict(X_test)
y_pred_svm = svm.predict(X_test_scaled)
```

In [22]:
```
def evaluate_model(y_true, y_pred, model_name):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred)
    recall = recall_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred)

    print(f"{model_name} Results:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1-score: {f1:.4f}\n")

evaluate_model(Y_test, y_pred_log_reg, "Logistic Regression")
evaluate_model(Y_test, y_pred_random_forest, "Random Forest")
evaluate_model(Y_test, y_pred_svm, "Support Vector Machine")
```

```
Logistic Regression Results:
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-score: 1.0000

Random Forest Results:
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-score: 1.0000

Support Vector Machine Results:
Accuracy: 0.9850
Precision: 0.0000
Recall: 0.0000
F1-score: 0.0000
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precis
ion is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control t
his behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```