TASK-2 (CREDIT CARD FRAUD DETECTION)

PROGRAM:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix, classification_report
from sklearn.impute import SimpleImputer

# Step 1: Load the dataset
file_path = r"/content/fraudTrain.csv"  # Update this path
data = pd.read_csv(file_path)
print(data.columns)

# Step 2: Data Preprocessing
# Separate features and target variable
# Identify non-numeric columns and drop them
non_numeric_columns =
data.select_dtypes(exclude=['number']).columns
X = data.drop(columns=['is_fraud','trans_date_trans_time'] +
list(non_numeric_columns))  # Features (all numeric columns except
'is_fraud' and 'trans_date_trans_time')
y = data['is_fraud']  # Target variable (fraudulent or legitimate)

# Handle missing values in features before scaling
imputer = SimpleImputer(strategy='mean') # Replace missing values
with the mean of each column
X_imputed = imputer.fit_transform(X)
```

```
# Standardize the feature values (important for models like
Logistic Regression)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)

# Handle missing values in the target variable (if any)
y.fillna(y.mode()[0], inplace=True)  # Fill NaN with the most
frequent value

# Step 3: Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42, stratify=y)

# Step 4: Train and Evaluate Models

# Logistic Regression
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train, y_train)
y_pred_log_reg = log_reg.predict(X_test)

# Decision Tree Classifier
dtree = DecisionTreeClassifier(random_state=42)
dtree.fit(X_train, y_train)
y_pred_dtree = dtree.predict(X_test)

# Random Forest Classifier
rf = RandomForestClassifier(random_state=42, n_estimators=100)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

# Step 5: Evaluate the models
def evaluate_model(y_test, y_pred, model_name):
    print(f"--- {model_name} ---")
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
    print(f"Precision: {precision_score(y_test, y_pred):.4f}")
    print(f"Recall: {recall_score(y_test, y_pred):.4f}")
```

```
    print(f"F1 Score: {f1_score(y_test, y_pred):.4f}")
    print("\nConfusion Matrix:")
    print(confusion_matrix(y_test, y_pred))
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))
    print("\n")

# Evaluate Logistic Regression
evaluate_model(y_test, y_pred_log_reg, "Logistic Regression")

# Evaluate Decision Tree
evaluate_model(y_test, y_pred_dtree, "Decision Tree")

# Evaluate Random Forest
evaluate_model(y_test, y_pred_rf, "Random Forest")
```

OUTPUT:

```
Index(['Unnamed: 0', 'trans_date_trans_time', 'cc_num', 'merchant', 'category',
       'amt', 'first', 'last', 'gender', 'street', 'city', 'state', 'zip',
       'lat', 'long', 'city_pop', 'job', 'dob', 'trans_num', 'unix_time',
       'merch_lat', 'merch_long', 'is_fraud'],
      dtype='object')
--- Logistic Regression ---
Accuracy: 0.9916
Precision: 0.0000
Recall: 0.0000
F1 Score: 0.0000

Confusion Matrix:
[[56922    48]
 [  433     0]]

Classification Report:
              precision    recall  f1-score   support

         0.0       0.99      1.00      1.00     56970
         1.0       0.00      0.00      0.00       433
```

```
   accuracy                         0.99     57403
  macro avg       0.50      0.50      0.50     57403
weighted avg       0.98      0.99      0.99     57403
```

```
--- Decision Tree ---
Accuracy: 0.9918
Precision: 0.4627
Recall: 0.5012
F1 Score: 0.4812

Confusion Matrix:
[[56718   252]
 [  216   217]]

Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     56970
         1.0       0.46      0.50      0.48       433

   accuracy                           0.99     57403
  macro avg       0.73      0.75      0.74     57403
weighted avg       0.99      0.99      0.99     57403
```

```
--- Random Forest ---
Accuracy: 0.9959
Precision: 0.8608
Recall: 0.5427
F1 Score: 0.6657

Confusion Matrix:
[[56932    38]
 [  198   235]]

Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     56970
         1.0       0.86      0.54      0.67       433
```

accuracy			**1.00**	**57403**
macro avg	**0.93**	**0.77**	**0.83**	**57403**
weighted avg	**1.00**	**1.00**	**1.00**	**57403**