TASK-3 (CUSTOMER CHURN PREDICTION)

PROGRAM:

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

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

# Inspect the dataset to understand its structure
print(data.head())
print(data.columns)

# Step 2: Data Preprocessing
# Drop unnecessary columns
data = data.drop(columns=['RowNumber', 'CustomerId', 'Surname'])

# Encode categorical variables
label_encoder = LabelEncoder()
data['Gender'] = label_encoder.fit_transform(data['Gender'])  # 0
for Female, 1 for Male
data['Geography'] = label_encoder.fit_transform(data['Geography'])
# Encode Geography

# Separate features and target variable
```

```
X = data.drop(columns=['Exited'])  # Features (all columns except
'Exited')
y = data['Exited']  # Target variable (churn: 1 if customer exited,
0 otherwise)

# Standardize the feature values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 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)

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

# Gradient Boosting Classifier
gb = GradientBoostingClassifier(random_state=42, n_estimators=100)
gb.fit(X_train, y_train)
y_pred_gb = gb.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 Random Forest
evaluate_model(y_test, y_pred_rf, "Random Forest")

# Evaluate Gradient Boosting
evaluate_model(y_test, y_pred_gb, "Gradient Boosting")
```

OUTPUT:

```
RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age
\
0          1    15634602  Hargrave          619    France  Female
42
1          2    15647311      Hill          608     Spain  Female
41
2          3    15619304      Onio          502    France  Female
42
3          4    15701354      Boni          699    France  Female
39
4          5    15737888  Mitchell          850     Spain  Female
43

   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1
```

```
   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore',
'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts',
'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
--- Logistic Regression ---
Accuracy: 0.8050
Precision: 0.5859
Recall: 0.1425
F1 Score: 0.2292

Confusion Matrix:
[[1552   41]
 [ 349   58]]

Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.97      0.89      1593
           1       0.59      0.14      0.23       407

    accuracy                           0.81      2000
   macro avg       0.70      0.56      0.56      2000
weighted avg       0.77      0.81      0.75      2000



--- Random Forest ---
Accuracy: 0.8640
Precision: 0.7848
Recall: 0.4570
F1 Score: 0.5776

Confusion Matrix:
[[1542   51]
```

```
 [ 221  186]]

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.97      0.92      1593
           1       0.78      0.46      0.58       407

    accuracy                           0.86      2000
   macro avg       0.83      0.71      0.75      2000
weighted avg       0.86      0.86      0.85      2000



--- Gradient Boosting ---
Accuracy: 0.8675
Precision: 0.7886
Recall: 0.4767
F1 Score: 0.5942

Confusion Matrix:
[[1541   52]
 [ 213  194]]

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.97      0.92      1593
           1       0.79      0.48      0.59       407

    accuracy                           0.87      2000
   macro avg       0.83      0.72      0.76      2000
weighted avg       0.86      0.87      0.85      2000
```