```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

# Load datasets
train_data = pd.read_csv('carl_insurance_train.csv')
test_data = pd.read_csv('carl_insurance_test.csv')

# Inspect the data
train_data_info = train_data.info()
test_data_info = test_data.info()

train_data_info, test_data_info
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 19 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Id                4000 non-null   int64
 1   Age               4000 non-null   int64
 2   Job               3981 non-null   object
 3   Marital           4000 non-null   object
 4   Education         3831 non-null   object
 5   Default           4000 non-null   int64
 6   Balance           4000 non-null   int64
 7   HHInsurance       4000 non-null   int64
 8   CarLoan           4000 non-null   int64
 9   Communication     3098 non-null   object
 10  LastContactDay    4000 non-null   int64
 11  LastContactMonth  4000 non-null   object
 12  NoOfContacts      4000 non-null   int64
 13  DaysPassed        4000 non-null   int64
 14  PrevAttempts      4000 non-null   int64
 15  Outcome           958 non-null    object
 16  CallStart         4000 non-null   object
 17  CallEnd           4000 non-null   object
 18  CarInsurance      4000 non-null   int64
dtypes: int64(11), object(8)
memory usage: 593.9+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
```

```
Data columns (total 19 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Id                1000 non-null   int64  
 1   Age               1000 non-null   int64  
 2   Job               995 non-null    object 
 3   Marital           1000 non-null   object 
 4   Education         953 non-null    object 
 5   Default           1000 non-null   int64  
 6   Balance           1000 non-null   int64  
 7   HHInsurance       1000 non-null   int64  
 8   CarLoan           1000 non-null   int64  
 9   Communication     779 non-null    object 
 10  LastContactDay    1000 non-null   int64  
 11  LastContactMonth  1000 non-null   object 
 12  NoOfContacts      1000 non-null   int64  
 13  DaysPassed        1000 non-null   int64  
 14  PrevAttempts      1000 non-null   int64  
 15  Outcome           243 non-null    object 
 16  CallStart         1000 non-null   object 
 17  CallEnd           1000 non-null   object 
 18  CarInsurance      0 non-null      float64
dtypes: float64(1), int64(10), object(8)
memory usage: 148.6+ KB

(None, None)
```

```
# Summary statistics for EDA
train_summary = train_data.describe(include='all')
test_summary = test_data.describe(include='all')
train_summary, test_summary
```

```
(                  Id          Age         Job  Marital  Education
Default  \
 count   4000.000000  4000.000000        3981     4000       3831
4000.000000
 unique          NaN          NaN          11        3          3
NaN
 top             NaN          NaN  management  married  secondary
NaN
 freq            NaN          NaN         893     2304       1988
NaN
 mean    2000.500000    41.214750         NaN      NaN        NaN
0.014500
 std     1154.844867    11.550194         NaN      NaN        NaN
0.119555
 min        1.000000    18.000000         NaN      NaN        NaN
0.000000
 25%     1000.750000    32.000000         NaN      NaN        NaN
0.000000
```

```
 50%     2000.500000    39.000000          NaN      NaN         NaN
0.000000
 75%     3000.250000    49.000000          NaN      NaN         NaN
0.000000
 max     4000.000000    95.000000          NaN      NaN         NaN
1.000000

               Balance  HHInsurance      CarLoan Communication
LastContactDay  \
 count    4000.000000   4000.00000  4000.000000          3098
4000.000000
 unique           NaN          NaN          NaN             2
NaN
 top              NaN          NaN          NaN      cellular
NaN
 freq             NaN          NaN          NaN          2831
NaN
 mean     1532.937250      0.49275     0.133000           NaN
15.721250
 std      3511.452489      0.50001     0.339617           NaN
8.425307
 min     -3058.000000      0.00000     0.000000           NaN
1.000000
 25%       111.000000      0.00000     0.000000           NaN
8.000000
 50%       551.500000      0.00000     0.000000           NaN
16.000000
 75%      1619.000000      1.00000     0.000000           NaN
22.000000
 max     98417.000000      1.00000     1.000000           NaN
31.000000

        LastContactMonth  NoOfContacts   DaysPassed  PrevAttempts
Outcome  \
 count              4000   4000.000000  4000.000000   4000.000000
958
 unique               12           NaN          NaN           NaN
3
 top                 may           NaN          NaN           NaN
failure
 freq               1049           NaN          NaN           NaN
437
 mean                NaN      2.607250    48.706500      0.717500
NaN
 std                 NaN      3.064204   106.685385      2.078647
NaN
 min                 NaN      1.000000    -1.000000      0.000000
NaN
 25%                 NaN      1.000000    -1.000000      0.000000
```

```
NaN
 50%                 NaN      2.000000    -1.000000      0.000000
NaN
 75%                 NaN      3.000000    -1.000000      0.000000
NaN
 max                 NaN     43.000000   854.000000     58.000000
NaN

        CallStart   CallEnd  CarInsurance
 count       4000      4000   4000.000000
 unique      3777      3764           NaN
 top     17:11:04  10:22:30           NaN
 freq           3         3           NaN
 mean         NaN       NaN      0.401000
 std          NaN       NaN      0.490162
 min          NaN       NaN      0.000000
 25%          NaN       NaN      0.000000
 50%          NaN       NaN      0.000000
 75%          NaN       NaN      1.000000
 max          NaN       NaN      1.000000  ,
                  Id          Age          Job  Marital  Education
Default  \
 count   1000.000000  1000.000000          995     1000        953
1000.000000
 unique          NaN          NaN           11        3          3
NaN
 top             NaN          NaN   management  married  secondary
NaN
 freq            NaN          NaN          221      594        501
NaN
 mean    4500.500000    41.473000          NaN      NaN        NaN
0.013000
 std      288.819436    12.051577          NaN      NaN        NaN
0.113331
 min     4001.000000    18.000000          NaN      NaN        NaN
0.000000
 25%     4250.750000    32.000000          NaN      NaN        NaN
0.000000
 50%     4500.500000    39.000000          NaN      NaN        NaN
0.000000
 75%     4750.250000    49.250000          NaN      NaN        NaN
0.000000
 max     5000.000000    92.000000          NaN      NaN        NaN
1.000000

              Balance  HHInsurance     CarLoan Communication
LastContactDay  \
 count    1000.000000  1000.000000  1000.00000           779
1000.000000
```

```
 unique            NaN           NaN         NaN             2
NaN
 top               NaN           NaN         NaN      cellular
NaN
 freq              NaN           NaN         NaN           698
NaN
 mean      1398.298000      0.513000     0.12100           NaN
15.546000
 std       2660.408024      0.500081     0.32629           NaN
8.370541
 min      -1980.000000      0.000000     0.00000           NaN
1.000000
 25%        114.750000      0.000000     0.00000           NaN
8.000000
 50%        517.500000      1.000000     0.00000           NaN
15.000000
 75%       1609.750000      1.000000     0.00000           NaN
21.000000
 max      41630.000000      1.000000     1.00000           NaN
31.000000

        LastContactMonth  NoOfContacts   DaysPassed  PrevAttempts
Outcome  \
 count              1000    1000.00000  1000.000000   1000.000000
243
 unique               12           NaN          NaN           NaN
3
 top                 may           NaN          NaN           NaN
failure
 freq                269           NaN          NaN           NaN
111
 mean                NaN       2.50800    51.653000      0.806000
NaN
 std                 NaN       2.57732   109.024855      2.034331
NaN
 min                 NaN       1.00000    -1.000000      0.000000
NaN
 25%                 NaN       1.00000    -1.000000      0.000000
NaN
 50%                 NaN       2.00000    -1.000000      0.000000
NaN
 75%                 NaN       3.00000    -1.000000      0.000000
NaN
 max                 NaN      34.00000   586.000000     20.000000
NaN

        CallStart   CallEnd  CarInsurance
 count       1000      1000           0.0
 unique       986       980           NaN
```

```
 top     09:11:24  11:27:58           NaN
 freq           2         2           NaN
 mean         NaN       NaN           NaN
 std          NaN       NaN           NaN
 min          NaN       NaN           NaN
 25%          NaN       NaN           NaN
 50%          NaN       NaN           NaN
 75%          NaN       NaN           NaN
 max          NaN       NaN           NaN  )

train_head = train_data.head()
test_head = test_data.head()

train_head, test_head

(   Id  Age          Job  Marital Education  Default  Balance
HHInsurance  \
 0   1   32   management   single  tertiary        0     1218
1
 1   2   32  blue-collar  married   primary        0     1156
1
 2   3   29   management   single  tertiary        0      637
1
 3   4   25      student   single   primary        0      373
1
 4   5   30   management  married  tertiary        0     2694
0

    CarLoan Communication  LastContactDay LastContactMonth
NoOfContacts  \
 0        0     telephone              28              jan
2
 1        0           NaN              26              may
5
 2        0      cellular               3              jun
1
 3        0      cellular              11              may
2
 4        0      cellular               3              jun
1

    DaysPassed  PrevAttempts  Outcome CallStart   CallEnd
CarInsurance
 0          -1             0      NaN  13:45:20  13:46:30
0
 1          -1             0      NaN  14:49:03  14:52:08
0
 2         119             1  failure  16:30:24  16:36:04
1
 3          -1             0      NaN  12:06:43  12:20:22
```

```
1
 4          -1             0      NaN  14:35:44  14:38:56
0  ,
      Id  Age          Job  Marital  Education  Default  Balance
HHInsurance  \
 0  4001   25       admin.   single  secondary        0        1
1
 1  4002   40  management  married   tertiary        0        0
1
 2  4003   44  management   single   tertiary        0    -1313
1
 3  4004   27    services   single  secondary        0     6279
1
 4  4005   53  technician  married  secondary        0     7984
1

    CarLoan Communication  LastContactDay LastContactMonth
NoOfContacts  \
 0        1           NaN              12              may
12
 1        1      cellular              24              jul
1
 2        1      cellular              15              may
10
 3        0      cellular               9              nov
1
 4        0      cellular               2              feb
1

    DaysPassed  PrevAttempts Outcome CallStart   CallEnd  CarInsurance

 0          -1             0     NaN  17:17:42  17:18:06           NaN

 1          -1             0     NaN  09:13:44  09:14:37           NaN

 2          -1             0     NaN  15:24:07  15:25:51           NaN

 3          -1             0     NaN  09:43:44  09:48:01           NaN

 4          -1             0     NaN  16:31:51  16:34:22           NaN
)
```

```
# Preprocessing

# Handle missing values
# Fill categorical columns with mode and numerical columns with median
categorical_cols = ['Job', 'Education', 'Communication', 'Outcome']
numerical_cols = ['Balance', 'DaysPassed']

for col in categorical_cols:
```

```
    train_data[col] = train_data[col].fillna(train_data[col].mode()
[0])
    test_data[col] = test_data[col].fillna(test_data[col].mode()[0])

# Fill numerical columns with median
for col in numerical_cols:
    train_data[col] = train_data[col].fillna(train_data[col].median())
    test_data[col] = test_data[col].fillna(test_data[col].median())

# Extract call duration in seconds from CallStart and CallEnd
def calculate_call_duration(start, end):
    start_time = pd.to_datetime(start, format='%H:%M:%S')
    end_time = pd.to_datetime(end, format='%H:%M:%S')
    return (end_time - start_time).dt.total_seconds()

train_data['CallDuration'] =
calculate_call_duration(train_data['CallStart'],
train_data['CallEnd'])
test_data['CallDuration'] =
calculate_call_duration(test_data['CallStart'], test_data['CallEnd'])

# Drop unnecessary columns
columns_to_drop = ['Id', 'CallStart', 'CallEnd']  # 'Id' is not
predictive, 'CallStart' and 'CallEnd' are processed
train_data.drop(columns=columns_to_drop, inplace=True)
test_data.drop(columns=columns_to_drop, inplace=True)

# Confirm changes
train_data.info(), test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 17 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Age               4000 non-null   int64
 1   Job               4000 non-null   object
 2   Marital           4000 non-null   object
 3   Education         4000 non-null   object
 4   Default           4000 non-null   int64
 5   Balance           4000 non-null   int64
 6   HHInsurance       4000 non-null   int64
 7   CarLoan           4000 non-null   int64
 8   Communication     4000 non-null   object
 9   LastContactDay    4000 non-null   int64
 10  LastContactMonth  4000 non-null   object
 11  NoOfContacts      4000 non-null   int64
 12  DaysPassed        4000 non-null   int64
 13  PrevAttempts      4000 non-null   int64
 14  Outcome           4000 non-null   object
```

```
 15  CarInsurance      4000 non-null   int64
 16  CallDuration      4000 non-null   float64
dtypes: float64(1), int64(10), object(6)
memory usage: 531.4+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Age               1000 non-null   int64
 1   Job               1000 non-null   object
 2   Marital           1000 non-null   object
 3   Education         1000 non-null   object
 4   Default           1000 non-null   int64
 5   Balance           1000 non-null   int64
 6   HHInsurance       1000 non-null   int64
 7   CarLoan           1000 non-null   int64
 8   Communication     1000 non-null   object
 9   LastContactDay    1000 non-null   int64
 10  LastContactMonth  1000 non-null   object
 11  NoOfContacts      1000 non-null   int64
 12  DaysPassed        1000 non-null   int64
 13  PrevAttempts      1000 non-null   int64
 14  Outcome           1000 non-null   object
 15  CarInsurance      0 non-null      float64
 16  CallDuration      1000 non-null   float64
dtypes: float64(2), int64(9), object(6)
memory usage: 132.9+ KB

(None, None)
```

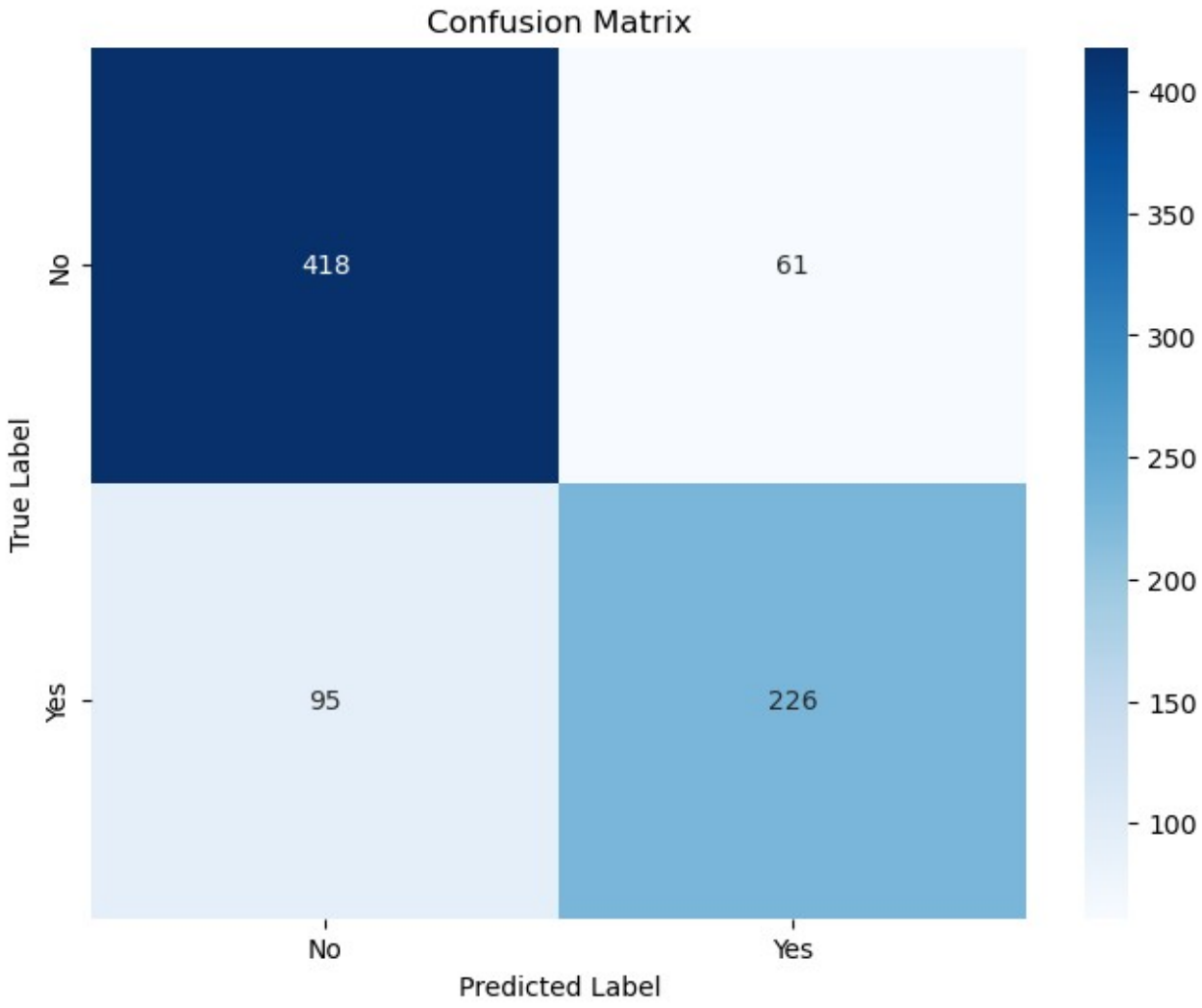
```
# Separate features and target variable for training
X = train_data.drop(columns=['CarInsurance'])
y = train_data['CarInsurance']

# Prepare column transformer for preprocessing
categorical_features = ['Job', 'Marital', 'Education',
'Communication', 'LastContactMonth', 'Outcome']
numerical_features = ['Age', 'Balance', 'LastContactDay',
'NoOfContacts', 'DaysPassed', 'PrevAttempts', 'CallDuration']


# Define the preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(handle_unknown='ignore'),
categorical_features)
    ]
)
```

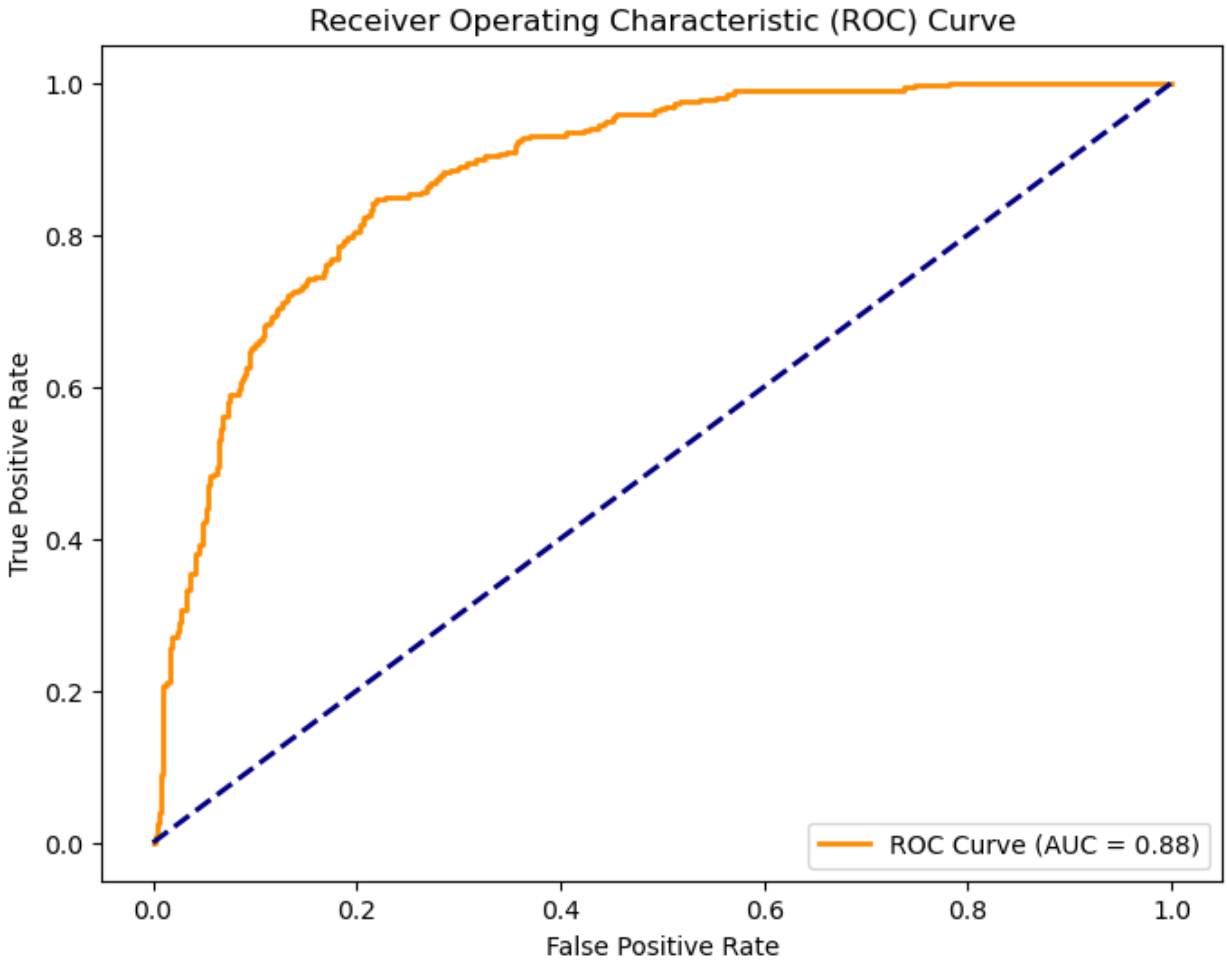
```
# Create a pipeline with Logistic Regression
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(random_state=42, max_iter=1000))
])

# Split the training data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)

# Train the pipeline
pipeline.fit(X_train, y_train)
```

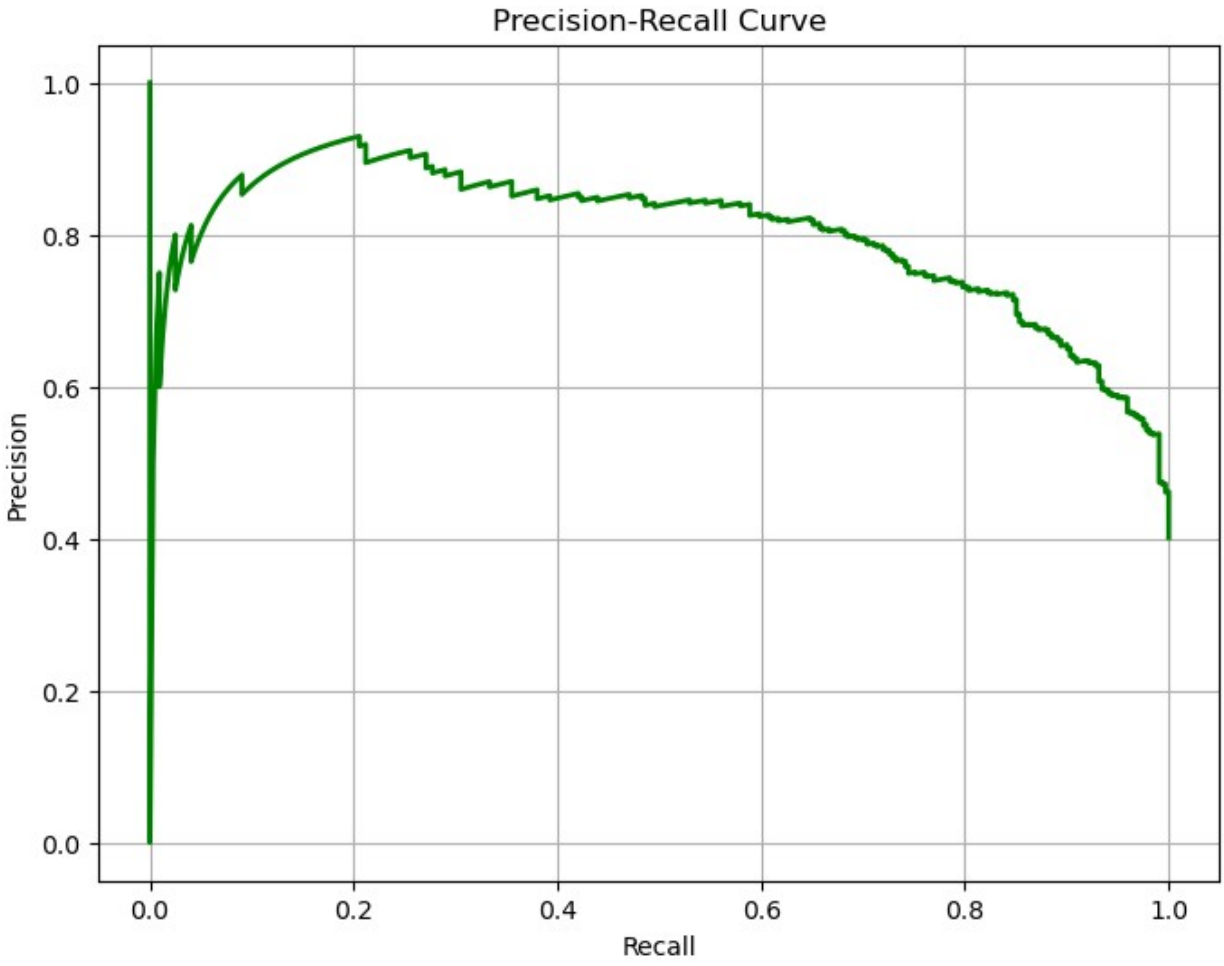
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
StandardScaler(),
                                                  ['Age', 'Balance',
                                                   'LastContactDay',
                                                   'NoOfContacts',
'DaysPassed',
                                                   'PrevAttempts',
                                                   'CallDuration']),
                                                 ('cat',

OneHotEncoder(handle_unknown='ignore'),
                                                  ['Job', 'Marital',
                                                   'Education',
'Communication',
                                                   'LastContactMonth',
                                                   'Outcome'])])),
                ('classifier',
                 LogisticRegression(max_iter=1000, random_state=42))])
```

```
# Validate the model
y_pred = pipeline.predict(X_val)
y_prob = pipeline.predict_proba(X_val)[:, 1]

# Predict probabilities and labels for the test dataset
test_predictions = pipeline.predict(test_data)
test_probabilities = pipeline.predict_proba(test_data)[:, 1]

# Combine predictions with test dataset for better context
test_data['PredictedLabel'] = test_predictions
test_data['PredictionProbability'] = test_probabilities

# Save the predictions to a CSV file
test_data[['PredictedLabel',
'PredictionProbability']].to_csv('test_predictions.csv', index=False)

print("Predictions have been saved to 'test_predictions.csv'.")
```

```
Predictions have been saved to 'test_predictions.csv'.

from sklearn.metrics import roc_curve, auc, precision_recall_curve

# Plot Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=['No', 'Yes'], yticklabels=['No', 'Yes'])
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

# ROC Curve
fpr, tpr, _ = roc_curve(y_val, y_prob)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f"ROC Curve (AUC =
{roc_auc:.2f})")
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic (ROC) Curve")
plt.legend(loc="lower right")
plt.show()

# Precision-Recall Curve
precision, recall, _ = precision_recall_curve(y_val, y_prob)

plt.figure(figsize=(8, 6))
plt.plot(recall, precision, color='green', lw=2)
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve")
plt.grid()
plt.show()
```


Confusion Matrix
418
61
95
226
No
Yes
True Label
No
Yes
Predicted Label
400
350
300
250
200
150
100


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC Curve (AUC = 0.88)


Precision-Recall Curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0