logistic

November 23, 2024

[6]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix

# Load the dataset
# Replace 'train_data.csv' and 'test_data.csv' with actual file paths
train_data = pd.read_csv('Logistictrain.csv')
test_data = pd.read_csv('Logistictest.csv')

# Display basic information
print("Training data shape:", train_data.shape)
print("Test data shape:", test_data.shape)
print(train_data.head())

# Check if columns exist before dropping
columns_to_drop = ['Id', 'CallStart', 'CallEnd']
for column in columns_to_drop:
    if column in train_data.columns:
        train_data = train_data.drop(columns=[column])
    if column in test_data.columns:
        test_data = test_data.drop(columns=[column])
```

```
Training data shape: (4000, 19)
Test data shape: (1000, 19)
   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
```

[10]:
```
# Handle missing values
# Fill missing values in categorical columns with 'unknown'
categorical_cols = ['Communication', 'Outcome']
for col in categorical_cols:
    train_data[col].fillna('unknown', inplace=True)
    test_data[col].fillna('unknown', inplace=True)

# Encode categorical variables
encode_cols = ['Job', 'Marital', 'Education', 'Communication', 'Outcome',␣
 ↪'LastContactMonth'] # Include 'LastContactMonth'
encoder = LabelEncoder()
for col in encode_cols:
    train_data[col] = encoder.fit_transform(train_data[col])
    test_data[col] = encoder.transform(test_data[col])

# Separate features and target
X_train = train_data.drop(columns=['CarInsurance'])
y_train = train_data['CarInsurance']
X_test = test_data.drop(columns=['CarInsurance'])

# Standardize numerical columns
# Select only numerical features for scaling
numerical_cols = X_train.select_dtypes(include=np.number).columns
scaler = StandardScaler()
X_train[numerical_cols] = scaler.fit_transform(X_train[numerical_cols])
X_test[numerical_cols] = scaler.transform(X_test[numerical_cols])

# Logistic Regression Model
logistic_model = LogisticRegression(random_state=42)
logistic_model.fit(X_train, y_train)

# Evaluate on training data
y_train_pred = logistic_model.predict(X_train)
print("Training Accuracy:", accuracy_score(y_train, y_train_pred))
print("Classification Report:\n", classification_report(y_train, y_train_pred))
```

```python
# Predict on test data
y_test_pred = logistic_model.predict(X_test)

# Save predictions
test_data['CarInsurance'] = y_test_pred
test_data[['CarInsurance']].to_csv('predictions.csv', index=False)

print("Predictions saved to 'predictions.csv'.")
```

```
<ipython-input-10-51adb12a5c83>:5: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  train_data[col].fillna('unknown', inplace=True)
<ipython-input-10-51adb12a5c83>:6: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  test_data[col].fillna('unknown', inplace=True)

Training Accuracy: 0.679
Classification Report:
               precision    recall  f1-score   support

           0       0.70      0.81      0.75      2396
           1       0.63      0.48      0.55      1604

    accuracy                           0.68      4000
   macro avg       0.67      0.65      0.65      4000
weighted avg       0.67      0.68      0.67      4000

Predictions saved to 'predictions.csv'.
```