In [1]:
```
#Experiment No: 10 - Create a Logistic regression model using housing datase
```

In [2]:
```
# Importing necessary libraries
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classificatior
```

In [3]:
```
# Setting the visual parameters
plt.rc("font", size=14)
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)

# Load the training and testing datasets using semicolon as the delimiter
train_df = pd.read_csv('train.csv', delimiter=';')
test_df = pd.read_csv('test.csv', delimiter=';')
```

In [4]:
```
# Display the first few rows of the training dataset
print("Training Data:\n", train_df.head())
print("\nTest Data:\n", test_df.head())

# Strip any leading/trailing whitespace from column names
train_df.columns = train_df.columns.str.strip()
test_df.columns = test_df.columns.str.strip()
```

```
Training Data:
    age           job  marital  education default  balance housing loan  \
0   58    management  married   tertiary      no     2143     yes   no
1   44    technician   single  secondary      no       29     yes   no
2   33  entrepreneur  married  secondary      no        2     yes  yes
3   47   blue-collar  married    unknown      no     1506     yes   no
4   33       unknown   single    unknown      no        1      no   no

   contact  day month  duration  campaign  pdays  previous poutcome   y
0  unknown    5   may       261         1     -1         0  unknown  no
1  unknown    5   may       151         1     -1         0  unknown  no
2  unknown    5   may        76         1     -1         0  unknown  no
3  unknown    5   may        92         1     -1         0  unknown  no
4  unknown    5   may       198         1     -1         0  unknown  no

Test Data:
    age          job  marital  education default  balance housing loan  \
0   30   unemployed  married    primary      no     1787      no   no
1   33     services  married  secondary      no     4789     yes  yes
2   35   management   single   tertiary      no     1350     yes   no
3   30   management  married   tertiary      no     1476     yes  yes
4   59  blue-collar  married  secondary      no        0     yes   no

    contact  day month  duration  campaign  pdays  previous poutcome   y
0  cellular   19   oct        79         1     -1         0  unknown  no
1  cellular   11   may       220         1    339         4  failure  no
2  cellular   16   apr       185         1    330         1  failure  no
3   unknown    3   jun       199         4     -1         0  unknown  no
4   unknown    5   may       226         1     -1         0  unknown  no
```

In [5]:

```python
# Check for null values
print("\nMissing values in Training Data:\n", train_df.isnull().sum())
print("\nMissing values in Test Data:\n", test_df.isnull().sum())
```

```
Missing values in Training Data:
 age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64

Missing values in Test Data:
 age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

In [6]:

```
# Encoding categorical variables
# Assuming 'y' is the target variable and its values are 'yes'/'no'
train_df['y'] = train_df['y'].map({'yes': 1, 'no': 0})
test_df['y'] = test_df['y'].map({'yes': 1, 'no': 0})

# Splitting the datasets into features and target variable
X_train = train_df.drop('y', axis=1)  # Features for training
y_train = train_df['y']                # Target variable for training

X_test = test_df.drop('y', axis=1)    # Features for testing
y_test = test_df['y']                  # Target variable for testing

# Convert categorical columns to dummy variables
X_train = pd.get_dummies(X_train, drop_first=True)
X_test = pd.get_dummies(X_test, drop_first=True)
```

In [7]:
```python
# Align the test set with the train set
X_test = X_test.reindex(columns=X_train.columns, fill_value=0)
```

In [8]:
```python
# Standardizing the features
scaler = preprocessing.StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [9]:
```python
# Creating the Logistic Regression model
model = LogisticRegression(max_iter=1000)
```

In [10]:
```python
# Fitting the model on the training data
model.fit(X_train, y_train)
```

Out[10]:
```
▾ LogisticRegression
LogisticRegression(max_iter=1000)
```

In [11]:
```python
# Making predictions on the test set
y_pred = model.predict(X_test)
```

In [12]:
```python
# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

In [13]:
```python
# Displaying the results
print("\nAccuracy:", accuracy)
print("\nConfusion Matrix:\n", conf_matrix)
print("\nClassification Report:\n", class_report)
```

```
Accuracy: 0.9022340190223402

Confusion Matrix:
 [[3905   95]
 [ 347  174]]

Classification Report:
               precision    recall  f1-score   support

           0       0.92      0.98      0.95      4000
           1       0.65      0.33      0.44       521

    accuracy                           0.90      4521
   macro avg       0.78      0.66      0.69      4521
weighted avg       0.89      0.90      0.89      4521
```

In [14]:
```python
# Visualizing the confusion matrix
plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
```

```
plt.title('Confusion Matrix')
plt.show()
```