Practical No. 07

Aim: To perform and analysis of Logistic Regression Algorithm

In [4]:

```
#Name : Taufiq Rafik Nagori
#Roll no. : 77 (BDA-B77)
#Section : B
#Subject : PE-II
```

In [8]:

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

In [10]:

```
os.getcwd()
```

Out[10]:

```
'C:\\Users\\USER'
```

In [12]:

```
os.chdir("C:\\Users\\USER\\Desktop")
```

In [16]:

```
data = pd.read_csv("HouseData.csv")
```

In [18]:

```
data.head()
```

Out[18]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4

In [20]:

```
data.tail()
```

Out[20]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	co
4595	2014-07-09 00:00:00	308166.666667	3.0	1.75	1510	6360	1.0	0	0	
4596	2014-07-09 00:00:00	534333.333333	3.0	2.50	1460	7573	2.0	0	0	
4597	2014-07-09 00:00:00	416904.166667	3.0	2.50	3010	7014	2.0	0	0	
4598	2014-07-10 00:00:00	203400.000000	4.0	2.00	2090	6630	1.0	0	0	
4599	2014-07-10 00:00:00	220600.000000	3.0	2.50	1490	8102	2.0	0	0	

In [22]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           4600 non-null   object
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64
 5   sqft_lot       4600 non-null   int64
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64
 8   view           4600 non-null   int64
 9   condition      4600 non-null   int64
 10  sqft_above     4600 non-null   int64
 11  sqft_basement  4600 non-null   int64
 12  yr_built       4600 non-null   int64
 13  yr_renovated   4600 non-null   int64
 14  street         4600 non-null   object
 15  city           4600 non-null   object
 16  statezip       4600 non-null   object
 17  country        4600 non-null   object
dtypes: float64(4), int64(9), object(5)
memory usage: 647.0+ KB
```

In [24]:

```
data.info
```

Out[24]:

```
<bound method DataFrame.info of                        date         price  bedrooms  bathr
ooms  sqft_living  \
0     2014-05-02 00:00:00  3.130000e+05       3.0       1.50         1340
1     2014-05-02 00:00:00  2.384000e+06       5.0       2.50         3650
2     2014-05-02 00:00:00  3.420000e+05       3.0       2.00         1930
3     2014-05-02 00:00:00  4.200000e+05       3.0       2.25         2000
4     2014-05-02 00:00:00  5.500000e+05       4.0       2.50         1940
...                   ...           ...       ...        ...          ...
4595  2014-07-09 00:00:00  3.081667e+05       3.0       1.75         1510
4596  2014-07-09 00:00:00  5.343333e+05       3.0       2.50         1460
4597  2014-07-09 00:00:00  4.169042e+05       3.0       2.50         3010
4598  2014-07-10 00:00:00  2.034000e+05       4.0       2.00         2090
4599  2014-07-10 00:00:00  2.206000e+05       3.0       2.50         1490

      sqft_lot  floors  waterfront  view  condition  sqft_above  \
0         7912     1.5           0     0          3        1340
1         9050     2.0           0     4          5        3370
2        11947     1.0           0     0          4        1930
3         8030     1.0           0     0          4        1000
4        10500     1.0           0     0          4        1140
...        ...     ...         ...   ...        ...         ...
4595      6360     1.0           0     0          4        1510
4596      7573     2.0           0     0          3        1460
4597      7014     2.0           0     0          3        3010
4598      6630     1.0           0     0          3        1070
4599      8102     2.0           0     0          4        1490

      sqft_basement  yr_built  yr_renovated                    street  \
0                 0      1955          2005      18810 Densmore Ave N
1               280      1921             0           709 W Blaine St
2                 0      1966             0  26206-26214 143rd Ave SE
3              1000      1963             0           857 170th Pl NE
4               800      1976          1992         9105 170th Ave NE
...             ...       ...           ...                       ...
4595              0      1954          1979           501 N 143rd St
4596              0      1983          2009          14855 SE 10th Pl
4597              0      2009             0          759 Ilwaco Pl NE
4598           1020      1974             0         5148 S Creston St
4599              0      1990             0         18717 SE 258th St

           city  statezip country
0     Shoreline  WA 98133     USA
1       Seattle  WA 98119     USA
2          Kent  WA 98042     USA
3      Bellevue  WA 98008     USA
4       Redmond  WA 98052     USA
...         ...       ...     ...
4595    Seattle  WA 98133     USA
4596   Bellevue  WA 98007     USA
4597     Renton  WA 98059     USA
4598    Seattle  WA 98178     USA
4599  Covington  WA 98042     USA

[4600 rows x 18 columns]>
```

In [26]:

```
data.shape
```

Out[26]:

```
(4600, 18)
```

In [28]:

```
data.size
```

Out[28]:

```
82800
```

In [30]:

```
data.ndim
```

Out[30]:

```
2
```

Data pre-processing, data-cleaning, mising value treatment

In [33]:

```
data.isna()
```

Out[33]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqf
0	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	
4595	False	False	False	False	False	False	False	False	False	False	
4596	False	False	False	False	False	False	False	False	False	False	
4597	False	False	False	False	False	False	False	False	False	False	
4598	False	False	False	False	False	False	False	False	False	False	
4599	False	False	False	False	False	False	False	False	False	False	

4600 rows × 18 columns

In [35]:

```
data.isna().any()
```

Out[35]:

```
date             False
price            False
bedrooms         False
bathrooms        False
sqft_living      False
sqft_lot         False
floors           False
waterfront       False
view             False
condition        False
sqft_above       False
sqft_basement    False
yr_built         False
```

```
yr_renovated     False
street           False
city             False
statezip         False
country          False
dtype: bool
```

In [37]:

```
data.isna().sum()
```

Out[37]:

```
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64
```

In [39]:

```
data
```

Out[39]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	con
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	
...	...	...	...	...	...	...	...	...	...	...
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	con
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	
4599	2014-07-10 00:00:00	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0	

4600 rows × 18 columns

In [41]:

```
data.isnull().sum()
```

Out[41]:

```
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64
```

In [43]:

```
data.columns
```

Out[43]:

```
Index(['date', 'price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
       'floors', 'waterfront', 'view', 'condition', 'sqft_above',
       'sqft_basement', 'yr_built', 'yr_renovated', 'street', 'city',
       'statezip', 'country'],
      dtype='object')
```

In [45]:

```
#feature selecting
data['expensive'] = np.where(data['price'] > data['price'].median(),1,0)
```

In [49]:

```
x = data.drop(['expensive','date','street','city','statezip','country'], axis=1)
y = data['expensive']
```

In [51]:

```
x
```

Out[51]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sq
0	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	3	
1	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	5	
2	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	4	
3	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	4	
4	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	4	
...	...	...	...	...	...	...	...	...	...	
4595	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	4	
4596	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	3	
4597	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	3	
4598	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	3	
4599	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0	4	

4600 rows × 13 columns

In [53]:

```
y
```

Out[53]:

```
0       0
1       1
2       0
3       0
4       1
       ..
4595    0
4596    1
4597    0
4598    0
4599    0
Name: expensive, Length: 4600, dtype: int32
```

In [55]:

```
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, random_state=42)
```

In [57]:

```
x_train
```

Out[57]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sq
1898	6.850000e+05	4.0	2.50	2770	45514	2.0	0	0	4	
1370	8.570000e+05	4.0	3.00	3720	29043	2.0	0	0	3	
3038	6.750000e+05	4.0	2.50	2810	11120	2.0	0	0	3	
2361	1.485000e+06	4.0	3.75	4030	10800	2.0	0	0	3	
156	5.610000e+05	3.0	2.00	2000	7000	2.0	0	0	3	
...	...	...	...	...	...	...	...	...	...	
4426	2.825089e+05	3.0	1.00	1180	5002	1.5	0	0	3	
466	8.750000e+05	4.0	2.00	2520	6000	1.0	0	0	3	
3092	2.900000e+05	3.0	1.00	1150	8145	1.0	0	0	4	
3772	6.000000e+05	2.0	1.00	910	2002	1.5	0	0	3	
860	4.750000e+05	2.0	1.00	1490	3825	1.0	0	0	3	

3680 rows × 13 columns

In [59]:

```
x_test
```

Out[59]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_
3683	544000.0	3.0	2.50	1460	1613	2.0	0	0	3	
4411	0.0	5.0	2.25	2000	7900	1.0	0	0	4	
2584	1712500.0	3.0	3.25	2940	5432	3.0	0	3	4	
69	365000.0	3.0	2.50	2200	7350	1.0	0	0	5	
1844	275000.0	3.0	2.50	1720	8755	1.0	0	0	3	
...	...	...	...	...	...	...	...	...	...	
1612	750000.0	3.0	1.75	1700	8400	1.0	0	0	3	
1068	230000.0	5.0	2.00	1930	6120	1.5	0	0	3	
4350	26590000.0	3.0	2.00	1180	7793	1.0	0	0	4	
3027	687000.0	4.0	2.50	2370	10083	2.0	0	0	5	
3455	289659.0	4.0	2.25	2260	7200	2.0	0	0	4	

920 rows × 13 columns

In [61]:

```
y_train
```

Out[61]:

```
1898    1
1370    1
3038    1
2361    1
```

```
156     1
       ..
4426    0
466     1
3092    0
3772    1
860     1
Name: expensive, Length: 3680, dtype: int32
```

In [63]:

```
y_test
```

Out[63]:

```
3683    1
4411    0
2584    1
69      0
1844    0
       ..
1612    1
1068    0
4350    1
3027    1
3455    0
Name: expensive, Length: 920, dtype: int32
```

In [65]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression().fit(x_train,y_train)
```

In [67]:

```
print("Train Accuracy:",model.score(x_train, y_train))
```

```
Train Accuracy: 0.9948369565217391
```

In [69]:

```
print("Test Accuracy:", model.score(x_test, y_test))
```

```
Test Accuracy: 0.9945652173913043
```

In [77]:

```
from sklearn.metrics import r2_score,mean_absolute_error
```

In [83]:

```
mean_absolute_error(y_test,y_predict)
```

Out[83]:

```
0.005434782608695652
```

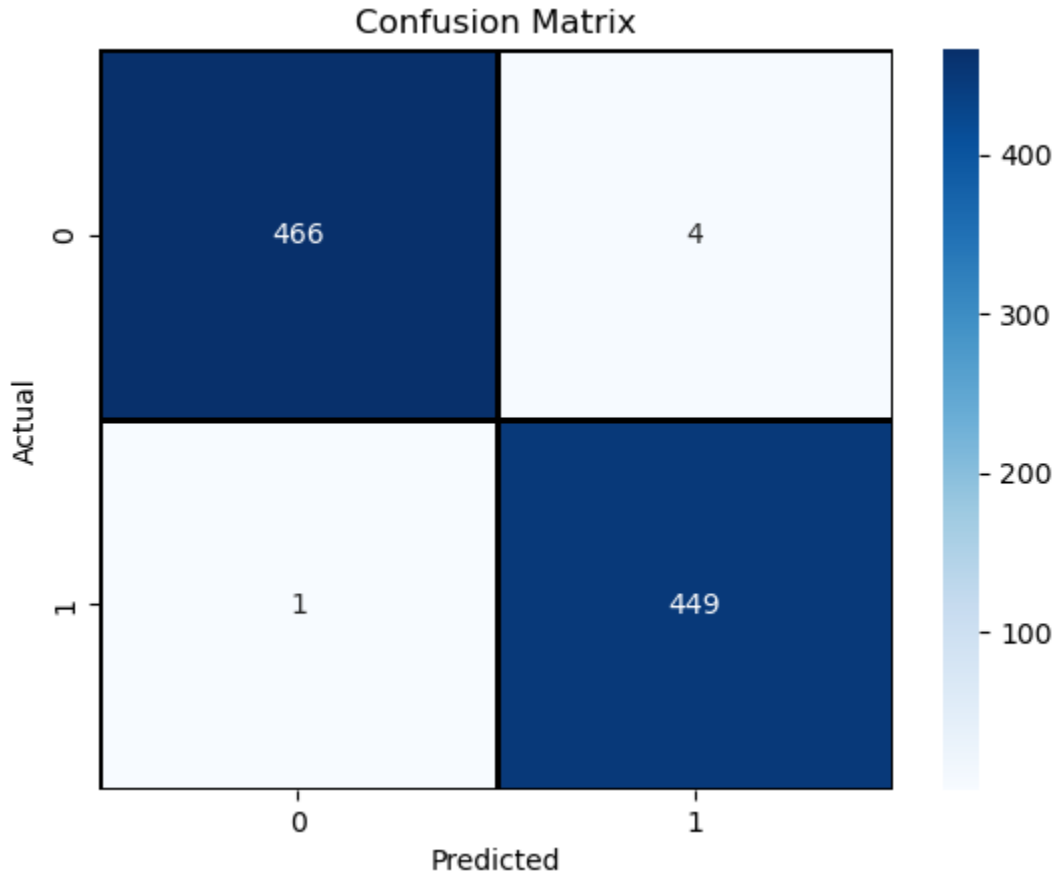
In [85]:

```
from sklearn.metrics import classification_report, confusion_matrix

y_predict = model.predict(x_test)

cm = confusion_matrix(y_test, y_predict)
sns.heatmap(cm, annot=True, cmap='Blues', fmt='d',linewidths=1, linecolor='black')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
print(classification_report(y_test, y_predict))
```




```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       470
           1       0.99      1.00      0.99       450

    accuracy                           0.99       920
   macro avg       0.99      0.99      0.99       920
weighted avg       0.99      0.99      0.99       920
```