Logistic Regression

October 30, 2021

1 Logistic Regression

1.1 Importing the libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
from sklearn.model_selection import train_test_split
```

[3]:
```
from sklearn.preprocessing import StandardScaler
```

[4]:
```
from sklearn.linear_model import LogisticRegression
```

[5]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.metrics import classification_report
```

1.2 Importing the dataset

[6]:
```
dataset = pd.read_csv("Social_Network_Ads.csv")
```

[7]:
```
x = dataset.iloc[: , :-1].values
y = dataset.iloc[: , -1].values
```

[8]:
```
dataset.head()
```

[8]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

[9]:
```
print(x)
```

```
[[    19  19000]
 [    35  20000]
 [    26  43000]
 [    27  57000]
```

```
[     19   76000]
[     27   58000]
[     27   84000]
[     32  150000]
[     25   33000]
[     35   65000]
[     26   80000]
[     26   52000]
[     20   86000]
[     32   18000]
[     18   82000]
[     29   80000]
[     47   25000]
[     45   26000]
[     46   28000]
[     48   29000]
[     45   22000]
[     47   49000]
[     48   41000]
[     45   22000]
[     46   23000]
[     47   20000]
[     49   28000]
[     47   30000]
[     29   43000]
[     31   18000]
[     31   74000]
[     27  137000]
[     21   16000]
[     28   44000]
[     27   90000]
[     35   27000]
[     33   28000]
[     30   49000]
[     26   72000]
[     27   31000]
[     27   17000]
[     33   51000]
[     35  108000]
[     30   15000]
[     28   84000]
[     23   20000]
[     25   79000]
[     27   54000]
[     30  135000]
[     31   89000]
[     24   32000]
[     18   44000]
```

```
[     29   83000]
[     35   23000]
[     27   58000]
[     24   55000]
[     23   48000]
[     28   79000]
[     22   18000]
[     32  117000]
[     27   20000]
[     25   87000]
[     23   66000]
[     32  120000]
[     59   83000]
[     24   58000]
[     24   19000]
[     23   82000]
[     22   63000]
[     31   68000]
[     25   80000]
[     24   27000]
[     20   23000]
[     33  113000]
[     32   18000]
[     34  112000]
[     18   52000]
[     22   27000]
[     28   87000]
[     26   17000]
[     30   80000]
[     39   42000]
[     20   49000]
[     35   88000]
[     30   62000]
[     31  118000]
[     24   55000]
[     28   85000]
[     26   81000]
[     35   50000]
[     22   81000]
[     30  116000]
[     26   15000]
[     29   28000]
[     29   83000]
[     35   44000]
[     35   25000]
[     28  123000]
[     35   73000]
[     28   37000]
```

```
[    27  88000]
[    28  59000]
[    32  86000]
[    33 149000]
[    19  21000]
[    21  72000]
[    26  35000]
[    27  89000]
[    26  86000]
[    38  80000]
[    39  71000]
[    37  71000]
[    38  61000]
[    37  55000]
[    42  80000]
[    40  57000]
[    35  75000]
[    36  52000]
[    40  59000]
[    41  59000]
[    36  75000]
[    37  72000]
[    40  75000]
[    35  53000]
[    41  51000]
[    39  61000]
[    42  65000]
[    26  32000]
[    30  17000]
[    26  84000]
[    31  58000]
[    33  31000]
[    30  87000]
[    21  68000]
[    28  55000]
[    23  63000]
[    20  82000]
[    30 107000]
[    28  59000]
[    19  25000]
[    19  85000]
[    18  68000]
[    35  59000]
[    30  89000]
[    34  25000]
[    24  89000]
[    27  96000]
[    41  30000]
```

```
[    29  61000]
[    20  74000]
[    26  15000]
[    41  45000]
[    31  76000]
[    36  50000]
[    40  47000]
[    31  15000]
[    46  59000]
[    29  75000]
[    26  30000]
[    32 135000]
[    32 100000]
[    25  90000]
[    37  33000]
[    35  38000]
[    33  69000]
[    18  86000]
[    22  55000]
[    35  71000]
[    29 148000]
[    29  47000]
[    21  88000]
[    34 115000]
[    26 118000]
[    34  43000]
[    34  72000]
[    23  28000]
[    35  47000]
[    25  22000]
[    24  23000]
[    31  34000]
[    26  16000]
[    31  71000]
[    32 117000]
[    33  43000]
[    33  60000]
[    31  66000]
[    20  82000]
[    33  41000]
[    35  72000]
[    28  32000]
[    24  84000]
[    19  26000]
[    29  43000]
[    19  70000]
[    28  89000]
[    34  43000]
```

```
[    30  79000]
[    20  36000]
[    26  80000]
[    35  22000]
[    35  39000]
[    49  74000]
[    39 134000]
[    41  71000]
[    58 101000]
[    47  47000]
[    55 130000]
[    52 114000]
[    40 142000]
[    46  22000]
[    48  96000]
[    52 150000]
[    59  42000]
[    35  58000]
[    47  43000]
[    60 108000]
[    49  65000]
[    40  78000]
[    46  96000]
[    59 143000]
[    41  80000]
[    35  91000]
[    37 144000]
[    60 102000]
[    35  60000]
[    37  53000]
[    36 126000]
[    56 133000]
[    40  72000]
[    42  80000]
[    35 147000]
[    39  42000]
[    40 107000]
[    49  86000]
[    38 112000]
[    46  79000]
[    40  57000]
[    37  80000]
[    46  82000]
[    53 143000]
[    42 149000]
[    38  59000]
[    50  88000]
[    56 104000]
```

```
[    41  72000]
[    51 146000]
[    35  50000]
[    57 122000]
[    41  52000]
[    35  97000]
[    44  39000]
[    37  52000]
[    48 134000]
[    37 146000]
[    50  44000]
[    52  90000]
[    41  72000]
[    40  57000]
[    58  95000]
[    45 131000]
[    35  77000]
[    36 144000]
[    55 125000]
[    35  72000]
[    48  90000]
[    42 108000]
[    40  75000]
[    37  74000]
[    47 144000]
[    40  61000]
[    43 133000]
[    59  76000]
[    60  42000]
[    39 106000]
[    57  26000]
[    57  74000]
[    38  71000]
[    49  88000]
[    52  38000]
[    50  36000]
[    59  88000]
[    35  61000]
[    37  70000]
[    52  21000]
[    48 141000]
[    37  93000]
[    37  62000]
[    48 138000]
[    41  79000]
[    37  78000]
[    39 134000]
[    49  89000]
```

```
[    55  39000]
[    37  77000]
[    35  57000]
[    36  63000]
[    42  73000]
[    43 112000]
[    45  79000]
[    46 117000]
[    58  38000]
[    48  74000]
[    37 137000]
[    37  79000]
[    40  60000]
[    42  54000]
[    51 134000]
[    47 113000]
[    36 125000]
[    38  50000]
[    42  70000]
[    39  96000]
[    38  50000]
[    49 141000]
[    39  79000]
[    39  75000]
[    54 104000]
[    35  55000]
[    45  32000]
[    36  60000]
[    52 138000]
[    53  82000]
[    41  52000]
[    48  30000]
[    48 131000]
[    41  60000]
[    41  72000]
[    42  75000]
[    36 118000]
[    47 107000]
[    38  51000]
[    48 119000]
[    42  65000]
[    40  65000]
[    57  60000]
[    36  54000]
[    58 144000]
[    35  79000]
[    38  55000]
[    39 122000]
```

```
[    53 104000]
[    35  75000]
[    38  65000]
[    47  51000]
[    47 105000]
[    41  63000]
[    53  72000]
[    54 108000]
[    39  77000]
[    38  61000]
[    38 113000]
[    37  75000]
[    42  90000]
[    37  57000]
[    36  99000]
[    60  34000]
[    54  70000]
[    41  72000]
[    40  71000]
[    42  54000]
[    43 129000]
[    53  34000]
[    47  50000]
[    42  79000]
[    42 104000]
[    59  29000]
[    58  47000]
[    46  88000]
[    38  71000]
[    54  26000]
[    60  46000]
[    60  83000]
[    39  73000]
[    59 130000]
[    37  80000]
[    46  32000]
[    46  74000]
[    42  53000]
[    41  87000]
[    58  23000]
[    42  64000]
[    48  33000]
[    44 139000]
[    49  28000]
[    57  33000]
[    56  60000]
[    49  39000]
[    39  71000]
```

```
[    47  34000]
[    48  35000]
[    48  33000]
[    47  23000]
[    45  45000]
[    60  42000]
[    39  59000]
[    46  41000]
[    51  23000]
[    50  20000]
[    36  33000]
[    49  36000]]
```

[10]:
```
print(y)
```

```
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 1
 1 1 0 0 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 0 1
 1 0 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0 0 0
 1 1 0 1 1 1 1 1 0 0 0 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0
 0 1 0 1 0 0 1 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 1 0 1
 1 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1]
```

1.3 Splitting the dataset into the Training set and Test set

[11]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.
↪25,random_state = 0)
```

1.4 Feature Scaling

[12]:
```
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

1.5 Training the Logistic Regression model on the Training dataset

[13]:
```
classifier = LogisticRegression(random_state = 0)
classifier.fit(x_train,y_train)
```

[13]:
```
LogisticRegression(random_state=0)
```

1.6 Predict New Result

[14]:
```
print(classifier.predict(sc.transform([[30,87000]])))
```

```
[0]
```

1.7 Predict Test Result

[15]:
```
y_predict = classifier.predict(x_test)
df=pd.DataFrame({'Actual':y_test, 'Predicted':y_predict})
pd.set_option('display.max_rows', df.shape[0]+1)
print(df)
```

```
    Actual  Predicted
0        0          0
1        0          0
2        0          0
3        0          0
4        0          0
5        0          0
6        0          0
7        1          1
8        0          0
9        0          1
10       0          0
11       0          0
12       0          0
13       0          0
14       0          0
15       0          0
16       0          0
17       0          0
18       1          1
19       0          0
20       0          0
21       1          1
22       0          0
23       1          1
24       0          0
25       1          1
26       0          0
27       0          0
28       0          0
29       0          0
30       0          0
31       1          0
32       1          1
33       0          0
34       0          0
```

35	0	0
36	0	0
37	0	0
38	0	0
39	1	1
40	0	0
41	0	0
42	0	0
43	0	0
44	1	1
45	0	0
46	0	0
47	1	1
48	0	0
49	1	1
50	1	1
51	0	0
52	0	0
53	0	0
54	1	1
55	1	0
56	0	0
57	0	0
58	1	0
59	0	0
60	0	0
61	1	1
62	0	0
63	1	0
64	0	0
65	1	1
66	0	0
67	0	0
68	0	0
69	0	0
70	1	1
71	0	0
72	0	0
73	1	0
74	0	0
75	0	0
76	0	1
77	0	0
78	1	1
79	1	1
80	1	1
81	0	1
82	0	0

```
83       0          0
84       1          1
85       1          1
86       0          0
87       1          1
88       1          0
89       0          0
90       0          0
91       1          1
92       0          0
93       0          0
94       0          0
95       1          0
96       0          0
97       1          0
98       1          1
99       1          1
```

1.8 Making The confusion Matrix and Evaluting model

[16]:
```
cm = confusion_matrix(y_test, y_predict)
print(cm)
accuracy_score(y_test,y_predict)
```

```
[[65  3]
 [ 8 24]]
```

[16]:
```
0.89
```

[17]:
```
report = classification_report(y_test, y_predict)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.89      0.96      0.92        68
           1       0.89      0.75      0.81        32

    accuracy                           0.89       100
   macro avg       0.89      0.85      0.87       100
weighted avg       0.89      0.89      0.89       100
```

1.9 Visualising the Training set result

[18]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(x_train), y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:,␣
 ↪0].max() + 10, step = 0.25),
```

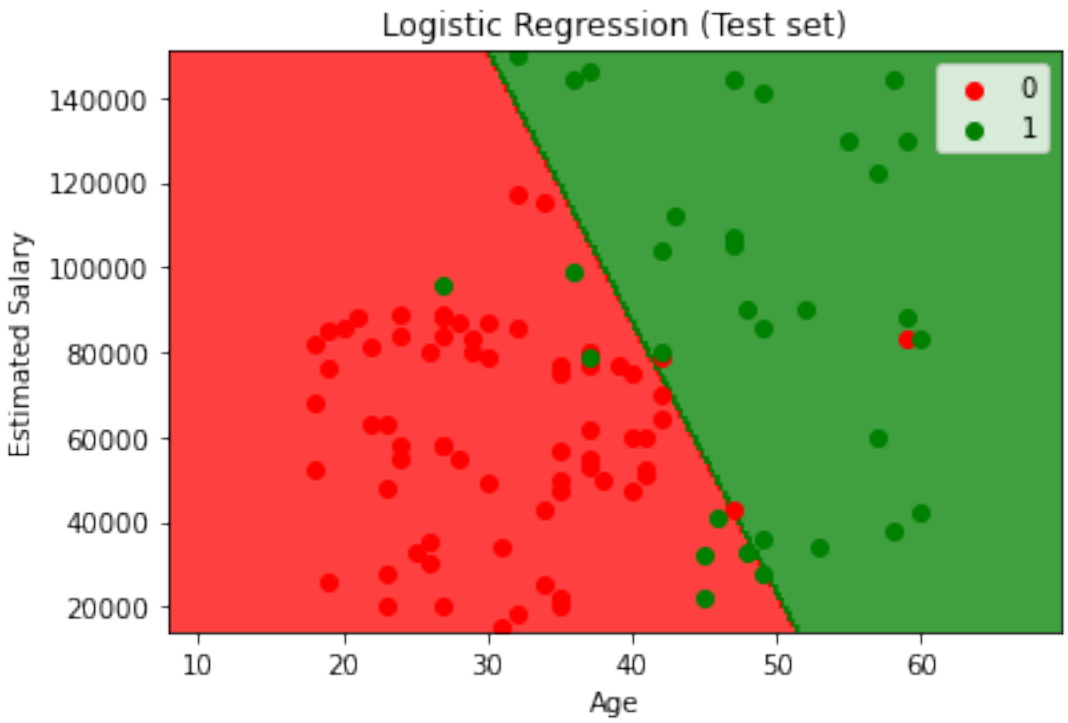
```
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:
 ↪, 1].max() + 1000, step = 0.25))
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.
 ↪ravel()]).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c =␣
 ↪ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with
a single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with
a single row if you intend to specify the same RGB or RGBA value for all points.
```

1.10 Visualising the Test result

[19]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(x_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:,␣
 ↪0].max() + 10, step = 0.25),
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:
 ↪, 1].max() + 1000, step = 0.25))
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.
 ↪ravel()]).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c =␣
 ↪ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with
a single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with
a single row if you intend to specify the same RGB or RGBA value for all points.
```


Logistic Regression (Test set)
Estimated Salary
Age
0
1
140000
120000
100000
80000
60000
40000
20000
10
20
30
40
50
60