Vikas-Logistic Regression Estimated Salary classification matrics

June 27, 2023

[24]:
```
##data manupulation
import pandas as pd
## data visuali
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

[25]:
```
data=pd.read_csv('C:/Users/Vikas/Downloads/SocialNetworkAds.csv')
```

[26]:
```
 data
```

[26]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
..	…	…	…	…	…
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

[400 rows x 5 columns]

[27]:
```
 ## i want to see the top 5 col
data.head(5)
```

[27]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

[28]:
```
# i want to see the botom 5 col
data.tail(5)
```

[28]:
```
     User ID  Gender  Age  EstimatedSalary  Purchased
395  15691863  Female   46            41000          1
396  15706071    Male   51            23000          1
397  15654296  Female   50            20000          1
398  15755018    Male   36            33000          0
399  15594041  Female   49            36000          1
```

[29]:
```
## i want to see in dataset all col
data.columns
```

[29]:
```
Index(['User ID', 'Gender', 'Age', 'EstimatedSalary', 'Purchased'],
dtype='object')
```

[30]:
```
## i want to see count of col all null values how much col and rows all␣
#information indataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

[31]:
```
## i want to see the max, min ,count,mean standard deviation.
data.describe().T
```

[31]:
```
                 count          mean           std         min           25%  \
User ID          400.0  1.569154e+07  71658.321581  15566689.0  15626763.75
Age              400.0  3.765500e+01     10.482877        18.0        29.75
EstimatedSalary  400.0  6.974250e+04  34096.960282     15000.0     43000.00
Purchased        400.0  3.575000e-01      0.479864         0.0         0.00

                      50%         75%         max
User ID          15694341.5  15750363.0  15815236.0
Age                 37.0        46.0        60.0
EstimatedSalary  70000.0     88000.0     150000.0
Purchased            0.0         1.0         1.0
```

[32]:
```
## i want to see the count of null values each col in dataset
data.isnull().sum()
```

[32]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

[33]:
```
## extracting the idependent variables
x=data.iloc[:,[2,3]].values
```

[34]:
```
## extract dependent variable
y=data.iloc[:,4].values
```

[35]:
```
 x
```

[35]:
```
array([[    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]], dtype=int64)
```

[36]:
```
y
```

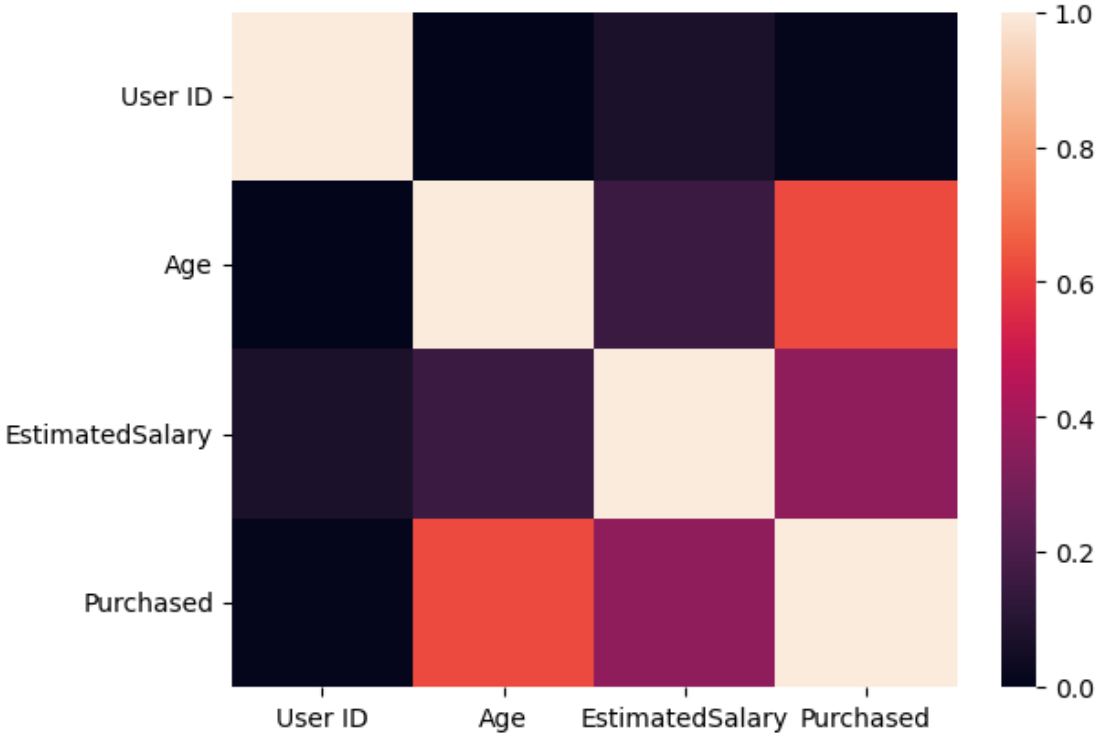
[36]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       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, 1, 0, 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, 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], dtype=int64)
```

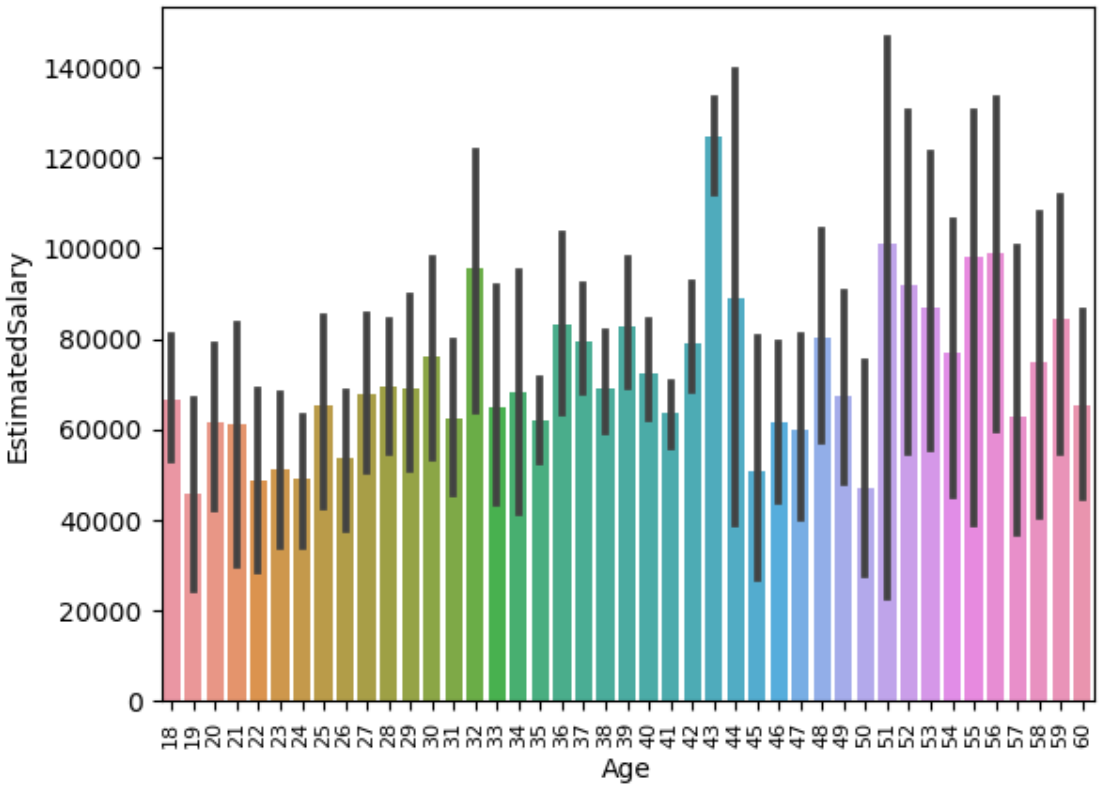
[37]:
```
## visualize the data
sns.heatmap(data.corr())
```

```
C:\Users\Vikas\AppData\Local\Temp\ipykernel_7304\907416350.py:2: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  sns.heatmap(data.corr())
```

[37]: <Axes: >



[38]:
```
## another way you visualize it,
sns.barplot(x='Age',y='EstimatedSalary',data=data)
plt.xticks(rotation='vertical',size=8)
plt.show()
```



[39]:
```
## spliting the data into training & testing set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=1/3,
random_state=0)
```

[40]:
```
 from sklearn.preprocessing import StandardScaler
sc_x=StandardScaler()
x_train=sc_x.fit_transform(x_train)
x_test=sc_x.transform(x_test)
```

[41]:
```
 ## fitting the Logistic regression to training dataset
from sklearn.linear_model import LogisticRegression
logr=LogisticRegression(random_state=0)
logr.fit(x_train,y_train)
```

[41]:
```
LogisticRegression(random_state=0)
```

[42]:
```
x_test
```

[42]:
```
array([[-0.76605872,  0.52531653],
       [ 0.02298918, -0.57354963],
       [-0.27290378,  0.16892751],
       [-0.76605872,  0.28772385],
       [-0.27290378, -0.57354963],
       [-1.06195168, -1.4645222 ],
       [-0.66742773, -1.61301763],
       [-0.1742728 ,  2.21816441],
       [-1.94963056, -0.03896609],
       [ 0.91066806, -0.78144323],
       [-0.76605872, -0.60324872],
       [-0.96332069, -0.42505421],
       [-0.07564181, -0.42505421],
       [ 0.12162016,  0.22832568],
       [-1.75236858,  0.49561745],
       [-0.56879674,  1.4162891 ],
       [-0.07564181,  0.22832568],
       [-1.85099957,  0.46591836],
       [ 1.69971595,  1.80237721],
       [-0.27290378, -1.40512403],
       [-0.27290378, -0.66264689],
       [ 0.91066806,  2.21816441],
       [ 0.31888214, -0.54385055],
       [ 0.91066806,  1.05990007],
       [-1.45647562, -1.22692951],
       [ 1.10793003,  2.12906715],
       [-0.96332069,  0.52531653],
       [-0.8646897 ,  0.31742293],
       [-0.07564181, -0.21716061],
       [-0.56879674,  0.49561745],
       [-1.6537376 ,  0.55501562],
       [-0.07564181,  0.28772385],
       [ 1.89697793, -0.27655878],
       [-0.07564181, -0.48445238],
       [-1.35784464, -0.33595695],
       [-1.94963056, -0.51415146],
       [-1.55510661,  0.34712202],
       [-0.37153477, -0.78144323],
       [-0.66742773, -1.048735  ],
       [ 1.10793003, -0.98933683],
       [-1.06195168,  0.55501562],
       [ 0.31888214, -0.51415146],
       [-1.06195168,  0.43621927],
```

```
[-0.27290378, -1.4645222 ],
[ 0.51614411,  1.26779367],
[-1.06195168, -0.33595695],
[-0.07564181,  0.31742293],
[ 1.40382299,  0.61441379],
[-1.16058266, -1.16753134],
[ 1.10793003,  0.49561745],
[ 1.89697793,  1.56478452],
[-0.37153477, -1.31602677],
[-0.27290378, -0.36565603],
[-0.37153477,  1.35689093],
[ 2.0942399 ,  0.55501562],
[ 0.71340608, -1.10813317],
[-0.8646897 ,  0.40652019],
[-1.16058266,  0.31742293],
[ 1.10793003, -1.22692951],
[-1.45647562, -1.4645222 ],
[-0.56879674, -1.52392037],
[ 2.19287089, -0.81114232],
[-1.85099957,  0.19862659],
[-0.1742728 ,  0.88170556],
[-1.85099957, -1.28632769],
[ 2.19287089,  0.40652019],
[-1.35784464,  0.5847147 ],
[-1.06195168, -0.33595695],
[ 0.22025115, -0.66264689],
[ 0.41751312,  0.02043208],
[-0.56879674,  2.39635892],
[-0.27290378,  0.22832568],
[-1.55510661, -0.18746152],
[ 0.71340608, -1.40512403],
[-1.06195168,  0.5847147 ],
[-1.94963056,  0.3768211 ],
[ 0.41751312,  0.28772385],
[ 0.22025115, -0.27655878],
[ 1.50245398, -1.048735  ],
[ 0.91066806,  1.11929824],
[ 1.99560891,  2.21816441],
[ 2.0942399 ,  0.40652019],
[-1.35784464, -0.42505421],
[-1.16058266, -1.01903592],
[ 1.99560891, -0.92993866],
[ 0.41751312,  0.31742293],
[ 0.22025115,  0.16892751],
[ 2.0942399 ,  1.80237721],
[ 0.81203707, -0.8408414 ],
[ 0.31888214, -0.27655878],
```

```
       [ 0.41751312, -0.15776244],
       [-0.07564181,  2.27756258],
       [-1.45647562, -0.6329478 ],
       [-1.25921365, -1.07843409],
       [-1.35784464,  0.43621927],
       [-1.06195168,  0.7926083 ],
       [-1.45647562, -0.18746152],
       [ 1.00929905, -1.07843409],
       [ 1.00929905,  0.61441379],
       [ 0.41751312,  1.03020099],
       [ 0.6147751 , -0.90023957],
       [-0.56879674,  1.50538635],
       [ 0.02298918, -0.57354963],
       [-0.56879674,  1.95087264],
       [ 1.40382299, -1.43482311],
       [ 1.50245398,  1.03020099],
       [ 0.12162016, -0.81114232],
       [ 0.02298918, -0.24685969],
       [-0.1742728 , -0.57354963],
       [-0.1742728 , -0.18746152],
       [-0.27290378, -1.31602677],
       [-0.27290378, -0.57354963],
       [ 0.41751312,  0.10952933],
       [ 0.91066806, -0.60324872],
       [ 2.0942399 , -1.19723043],
       [ 1.10793003, -0.12806335],
       [ 0.71340608,  1.83207629],
       [-0.66742773,  0.5847147 ],
       [ 0.81203707,  0.3768211 ],
       [ 0.91066806, -0.54385055],
       [-1.16058266, -1.61301763],
       [ 2.19287089,  0.97080281],
       [ 0.02298918,  1.26779367],
       [ 0.22025115,  1.11929824],
       [ 0.41751312, -0.48445238],
       [-0.27290378, -0.30625786],
       [ 1.00929905, -0.8408414 ],
       [ 1.00929905,  1.92117355],
       [ 0.02298918,  1.29749276],
       [-0.8646897 ,  2.33696075],
       [-1.16058266, -1.61301763],
       [ 2.19287089, -0.81114232],
       [-1.35784464, -1.49422128],
       [ 0.41751312,  2.36665983]])
```

[44]:
```
y_pred=logr.predict(x_test)
```

[46]:
```
y_pred
```

[46]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1,
       0, 1], dtype=int64)
```

[49]:
```
## here i identityfy is purchased or not predicted
logr.predict_proba(x_test)
```

[49]:
```
array([[0.89124462, 0.10875538],
       [0.82680904, 0.17319096],
       [0.80911097, 0.19088903],
       [0.9117147 , 0.0882853 ],
       [0.89723339, 0.10276661],
       [0.99047055, 0.00952945],
       [0.98171848, 0.01828152],
       [0.32051853, 0.67948147],
       [0.99374071, 0.00625929],
       [0.48862212, 0.51137788],
       [0.96090525, 0.03909475],
       [0.96865676, 0.03134324],
       [0.83477888, 0.16522112],
       [0.64142004, 0.35857996],
       [0.98439871, 0.01560129],
       [0.69718585, 0.30281415],
       [0.72789799, 0.27210201],
       [0.98756598, 0.01243402],
       [0.01521694, 0.98478306],
       [0.95148637, 0.04851363],
       [0.90495794, 0.09504206],
       [0.04903838, 0.95096162],
       [0.71719658, 0.28280342],
       [0.13733582, 0.86266418],
       [0.99460815, 0.00539185],
       [0.03624247, 0.96375753],
       [0.9245592 , 0.0754408 ],
       [0.92463629, 0.07536371],
       [0.80495418, 0.19504582],
       [0.84940874, 0.15059126],
       [0.97987856, 0.02012144],
       [0.71629862, 0.28370138],
       [0.07248358, 0.92751642],
       [0.84259855, 0.15740145],
```

```
[0.98446649, 0.01553351],
[0.99604921, 0.00395079],
[0.97990035, 0.02009965],
[0.92893292, 0.07106708],
[0.96875732, 0.03124268],
[0.43889714, 0.56110286],
[0.93573336, 0.06426664],
[0.71129739, 0.28870261],
[0.94234572, 0.05765428],
[0.95408625, 0.04591375],
[0.22530305, 0.77469695],
[0.97195295, 0.02804705],
[0.71038826, 0.28961174],
[0.08240145, 0.91759855],
[0.98960493, 0.01039507],
[0.15566245, 0.84433755],
[0.01285318, 0.98714682],
[0.95650719, 0.04349281],
[0.87702175, 0.12297825],
[0.61993932, 0.38006068],
[0.02273335, 0.97726665],
[0.66259518, 0.33740482],
[0.91836683, 0.08163317],
[0.95733405, 0.04266595],
[0.49639921, 0.50360079],
[0.99571652, 0.00428348],
[0.97576621, 0.02423379],
[0.06707292, 0.93292708],
[0.9903867 , 0.0096133 ],
[0.63397575, 0.36602425],
[0.99771744, 0.00228256],
[0.02150711, 0.97849289],
[0.96278298, 0.03721702],
[0.97195295, 0.02804705],
[0.77685663, 0.22314337],
[0.54492541, 0.45507459],
[0.47005721, 0.52994279],
[0.80002249, 0.19997751],
[0.9879555 , 0.0120445 ],
[0.72390705, 0.27609295],
[0.9339731 , 0.0660269 ],
[0.99064779, 0.00935221],
[0.48002248, 0.51997752],
[0.70509583, 0.29490417],
[0.27037274, 0.72962726],
[0.13062939, 0.86937061],
[0.00560569, 0.99439431],
```

```
[0.02617558, 0.97382442],
[0.98573831, 0.01426169],
[0.98800801, 0.01199199],
[0.10769761, 0.89230239],
[0.47281257, 0.52718743],
[0.60781008, 0.39218992],
[0.0068617 , 0.9931383 ],
[0.5531769 , 0.4468231 ],
[0.66160565, 0.33839435],
[0.58749378, 0.41250622],
[0.26689682, 0.73310318],
[0.99042872, 0.00957128],
[0.99071923, 0.00928077],
[0.96762895, 0.03237105],
[0.92034576, 0.07965424],
[0.98531021, 0.01468979],
[0.51057336, 0.48942664],
[0.16724846, 0.83275154],
[0.30947726, 0.69052274],
[0.66234793, 0.33765207],
[0.67857251, 0.32142749],
[0.82680904, 0.17319096],
[0.57777651, 0.42222349],
[0.3975345 , 0.6024655 ],
[0.04670626, 0.95329374],
[0.83106433, 0.16893567],
[0.77647303, 0.22352697],
[0.87714096, 0.12285904],
[0.83059808, 0.16940192],
[0.9473233 , 0.0526767 ],
[0.89723339, 0.10276661],
[0.52335154, 0.47664846],
[0.44548107, 0.55451893],
[0.11349147, 0.88650853],
[0.2527746 , 0.7472254 ],
[0.10095471, 0.89904529],
[0.86347808, 0.13652192],
[0.27456806, 0.72543194],
[0.43124948, 0.56875052],
[0.99323726, 0.00676274],
[0.01253255, 0.98746745],
[0.44302406, 0.55697594],
[0.38064141, 0.61935859],
[0.66185317, 0.33814683],
[0.87064968, 0.12935032],
[0.45290602, 0.54709398],
[0.05329972, 0.94670028],
```

```
       [0.43590399, 0.56409601],
       [0.63217776, 0.36782224],
       [0.99323726, 0.00676274],
       [0.06707292, 0.93292708],
       [0.99491506, 0.00508494],
       [0.10877986, 0.89122014]])
```

[50]:
```
## i want to see score
logr.score(x_test,y_test)
```

[50]:
```
0.8731343283582089
```

[64]:
```
#3 confusion matrics
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix : \n", cm)
```

```
Confusion Matrix :
 [[79  6]
 [11 38]]
```

[67]:
```
# caluclate the accuracy
from sklearn.metrics import accuracy_score

print ("Accuracy : ", accuracy_score(y_test, y_pred))
```

```
Accuracy :  0.8731343283582089
```

[68]:
```
## i want to see the errors
from sklearn import metrics
print('MEA:',metrics.mean_absolute_error(y_test,y_pred))
print('MSE:',metrics.mean_squared_error(y_test,y_pred))
print('RMSE:',np.sqrt(metrics.mean_absolute_error(y_test,y_pred)))
```

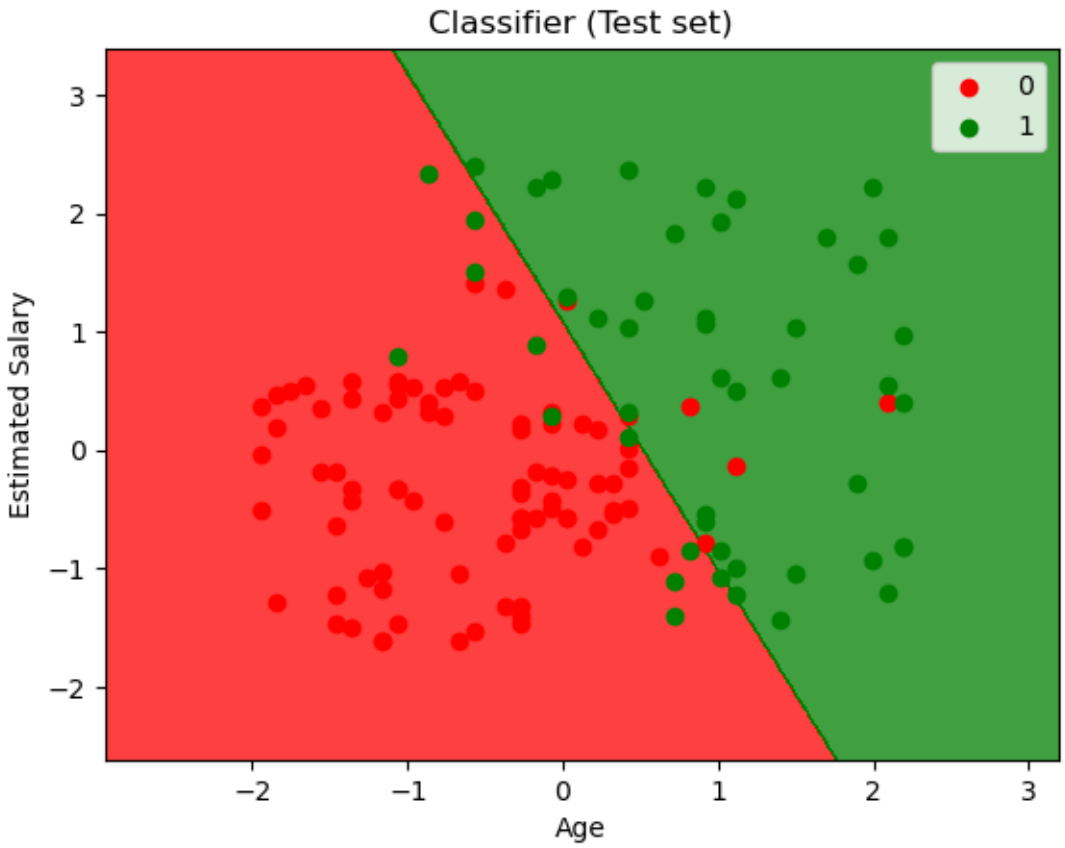
```
MEA: 0.12686567164179105
MSE: 0.12686567164179105
RMSE: 0.35618207653079775
```

[71]:
```
import numpy as np
print("Mean of y_test: ", np.mean(y_test))
print("Mean of y_pred: ", np.mean(y_pred))

print("Median of y_test: ", np.median(y_test))
print("Median of y_pred: ", np.median(y_pred))
```

```
Mean of y_test:  0.3656716417910448
Mean of y_pred:  0.3283582089552239
Median of y_test:  0.0
Median of y_pred:  0.0
```

[75]:
```
from matplotlib.colors import ListedColormap

x_set, y_set = x_test, y_test
x1, x2 = np.meshgrid(np.arange(start = x_set[:, 0].min() - 1,
                               stop = x_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = x_set[:, 1].min() - 1,
                               stop = x_set[:, 1].max() + 1, step = 0.01))

plt.contourf(x1, x2, logr.predict(
             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('Classifier (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
C:\Users\Vikas\AppData\Local\Temp\ipykernel_7304\102137657.py:17: UserWarning:
*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 2D array with a
single row if you intend to specify the same RGB or RGBA value for all points.
  plt.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
```


Classifier (Test set)
Estimated Salary
Age
0
1
3
2
1
0
−1
−2
−2
−1
0
1
2
3

[ ]: