Importing Libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from matplotlib.colors import ListedColormap
from sklearn import metrics
import seaborn as sns
%matplotlib inline
import warnings;
warnings.filterwarnings('ignore');
```

Importing Dataset

```
dataset=pd.read_csv('Social_Network_Ads.csv')
dataset
```

```
      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]
```

Extracting Variables

```
X = dataset.iloc[:,[2,3]].values
X
```

```
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],
```

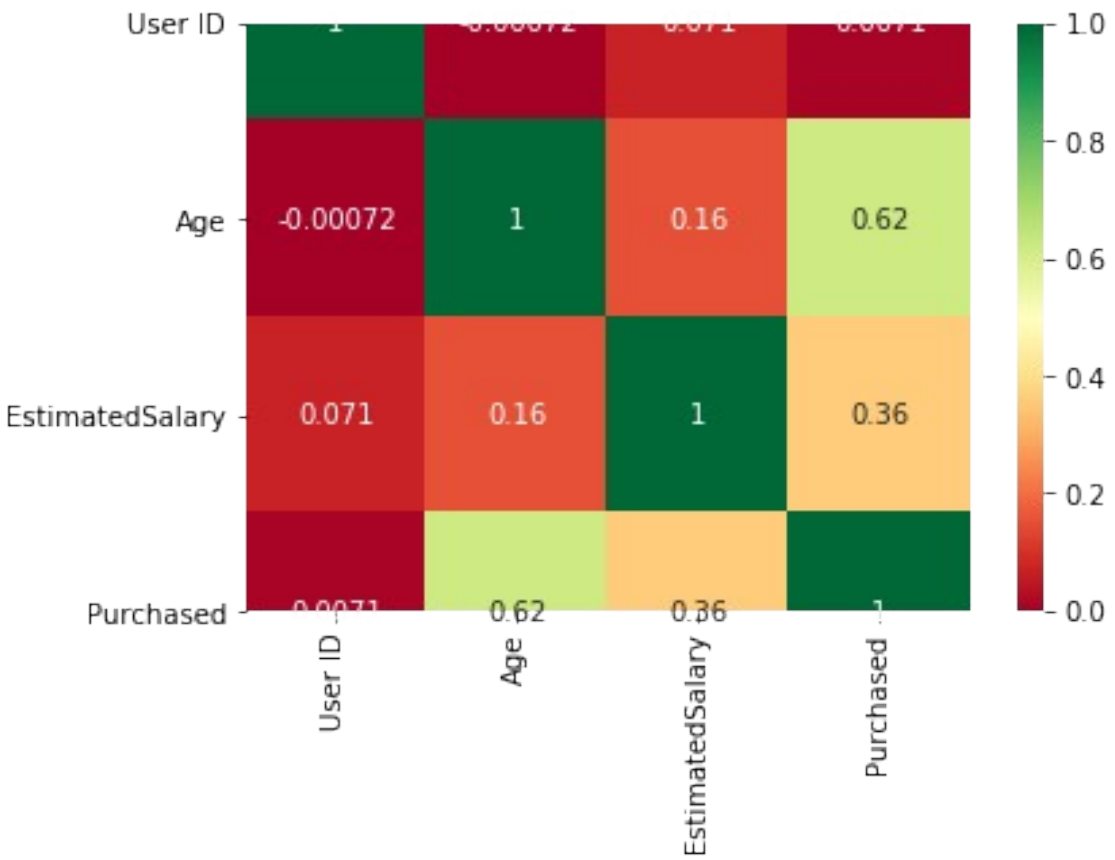
```
       [    36,  99000],
       [    60,  34000],
       [    54,  70000],
       [    41,  72000],
       [    40,  71000],
       [    42,  54000],
       [    43, 129000],
       [    53,  34000],
       [    47,  50000],
       [    42,  79000],
       [    42, 104000],
       [    59,  29000],
       [    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]])

Y = dataset.iloc[:,4].values
Y
```

```
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, 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])
```

Heatmap to see correlation

```
sns.heatmap(dataset.corr(), annot = True, cmap = 'RdYlGn')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff142682910>
```



Train Test Split

```
X_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=1/3,rando
m_state=32)
```

Feature Scaling

```
standard_Scaler=StandardScaler()
X_train = standard_Scaler.fit_transform(X_train)
x_test = standard_Scaler.transform(x_test)

x_test
```

```
array([[ 2.2508008 ,  0.96423267],
       [-1.3224201 ,  0.57131297],
       [-1.02465169,  0.78288512],
       [ 2.05228853, -0.69811991],
       [ 0.96047103,  2.23366555],
       [ 0.06716581,  0.29929164],
       [-0.32985874, -1.36306094],
       [ 0.96047103,  1.05490645],
       [ 1.95303239, -1.1212642 ],
       [-0.82613942, -0.27497561],
       [ 0.86121489, -1.45373471],
       [-1.71944464,  0.11794409],
```

```
       [ 2.05228853, -0.97014124],
       [-1.81870078, -0.00295428],
       [-1.42167623, -0.66789531],
       [ 0.26567808, -0.3354248 ],
       [-1.02465169, -0.36564939],
       [-0.13134647,  2.23366555],
       [-1.12390783, -1.60485767],
       [ 1.85377625,  1.02468185],
       [ 1.45675171, -1.48395931],
       [-0.52837101, -1.57463308],
       [ 1.1589833 ,  2.14299177],
       [ 1.25823944,  0.54108838],
       [ 0.36493421,  0.0574949 ],
       [ 0.96047103, -0.81901828],
       [-0.2306026 , -1.36306094],
       [-0.32985874, -0.81901828],
       [ 1.1589833 ,  0.48063919],
       [ 1.05972717, -1.24216257],
       [ 2.15154466, -0.84924287],
       [ 0.16642194,  1.93141962],
       [-1.81870078,  0.4504146 ],
       [ 1.45675171,  0.60153756],
       [ 0.26567808, -0.27497561],
       [ 0.26567808,  0.14816868],
       [-1.02465169, -1.18171338],
       [ 1.05972717,  2.05231799],
       [-0.2306026 , -0.97014124],
       [-0.72688328, -0.63767072],
       [ 0.36493421,  0.26906705],
       [-0.92539555,  0.42019001],
       [ 1.55600785, -1.09103961],
       [-1.02465169, -1.5141839 ],
       [-1.02465169,  0.60153756],
       [ 2.2508008 , -0.7283445 ],
       [ 1.35749557,  2.29411473],
       [-0.92539555, -0.78879368],
       [ 0.96047103, -0.69811991],
       [-1.22316396,  0.29929164],
       [-0.42911487, -0.30520021],
       [ 1.25823944, -1.5141839 ],
       [-0.82613942, -0.81901828],
       [-0.2306026 , -0.39587398],
       [-0.52837101,  0.48063919],
       [-1.3224201 , -1.54440849],
       [ 1.05972717,  1.84074585],
       [-1.71944464, -1.03059042],
       [ 1.55600785,  0.0574949 ],
       [-0.03209033, -0.00295428],
       [-1.42167623, -1.5141839 ],
```

```
       [ 0.26567808, -0.15407724],
       [-1.02465169, -1.60485767],
       [ 1.05972717,  0.78288512],
       [ 1.1589833 ,  0.54108838],
       [-0.2306026 , -0.78879368],
       [ 0.46419035,  0.14816868],
       [ 0.96047103,  1.29670318],
       [-1.12390783, -1.15148879],
       [ 0.86121489,  0.35974082],
       [-1.02465169,  2.0220934 ],
       [-1.12390783,  0.29929164],
       [ 1.1589833 ,  0.57131297],
       [ 1.85377625, -0.30520021],
       [ 0.76195876,  0.26906705],
       [-0.03209033,  2.29411473],
       [-1.12390783,  0.48063919],
       [ 0.36493421, -0.75856909],
       [-1.02465169,  0.57131297],
       [ 0.46419035, -0.48654776],
       [ 0.16642194,  0.08771949],
       [ 1.45675171, -0.97014124],
       [ 1.95303239,  1.56872452],
       [-1.22316396,  0.26906705],
       [ 1.1589833 ,  0.11794409],
       [ 1.65526398,  1.14558022],
       [-0.72688328,  1.11535563],
       [ 0.66270262,  2.08254258],
       [ 2.2508008 , -1.09103961],
       [ 0.86121489, -1.15148879],
       [ 1.1589833 , -1.27238716],
       [-0.2306026 , -1.5141839 ],
       [-1.71944464,  0.48063919],
       [-1.52093237, -1.57463308],
       [-0.2306026 ,  0.20861786],
       [-0.82613942,  0.38996542],
       [ 0.06716581, -0.15407724],
       [-1.12390783, -1.63508227],
       [-0.13134647, -0.21452643],
       [-0.82613942, -0.69811991],
       [ 0.46419035,  0.08771949],
       [ 0.86121489, -1.27238716],
       [-0.92539555, -0.3354248 ],
       [ 0.76195876, -1.45373471],
       [-0.03209033,  0.29929164],
       [-1.71944464,  0.35974082],
       [ 2.15154466,  0.17839327],
       [ 2.15154466, -1.24216257],
       [ 0.06716581, -0.60744613],
       [-0.03209033, -0.39587398],
```

```
       [ 0.06716581, -0.27497561],
       [-0.2306026 ,  2.32433932],
       [-0.03209033,  2.0220934 ],
       [-0.2306026 ,  0.0574949 ],
       [-1.02465169, -0.39587398],
       [ 0.06716581,  0.02727031],
       [-0.52837101,  1.50827533],
       [ 0.96047103, -1.42351012],
       [-1.22316396, -1.45373471],
       [ 0.96047103, -1.21193797],
       [ 0.36493421, -0.57722154],
       [-0.42911487,  2.38478851],
       [-1.12390783,  1.44782615],
       [-0.2306026 , -1.45373471],
       [-1.91795691,  0.35974082],
       [-0.03209033,  0.29929164],
       [-0.52837101,  1.41760155],
       [ 1.65526398,  1.02468185],
       [-0.13134647,  1.65939829],
       [ 1.25823944, -0.78879368],
       [-1.6201885 ,  0.54108838],
       [-0.72688328,  0.29929164],
       [-0.82613942, -0.81901828],
       [ 0.26567808, -0.39587398]])
```

Instantiating and fitting the model to training Dataset

```
log_reg=LogisticRegression(random_state=0)
log_reg.fit(X_train,y_train)

LogisticRegression(C=1.0, class_weight=None, dual=False,
fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=0, solver='warn', tol=0.0001,
verbose=0,
                   warm_start=False)
```

Prediction for Test Dataset

```
y_pred=log_reg.predict(x_test)
y_pred

array([1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1,
0,
       1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
0,
       0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1,
0,
       1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0,
1,
```

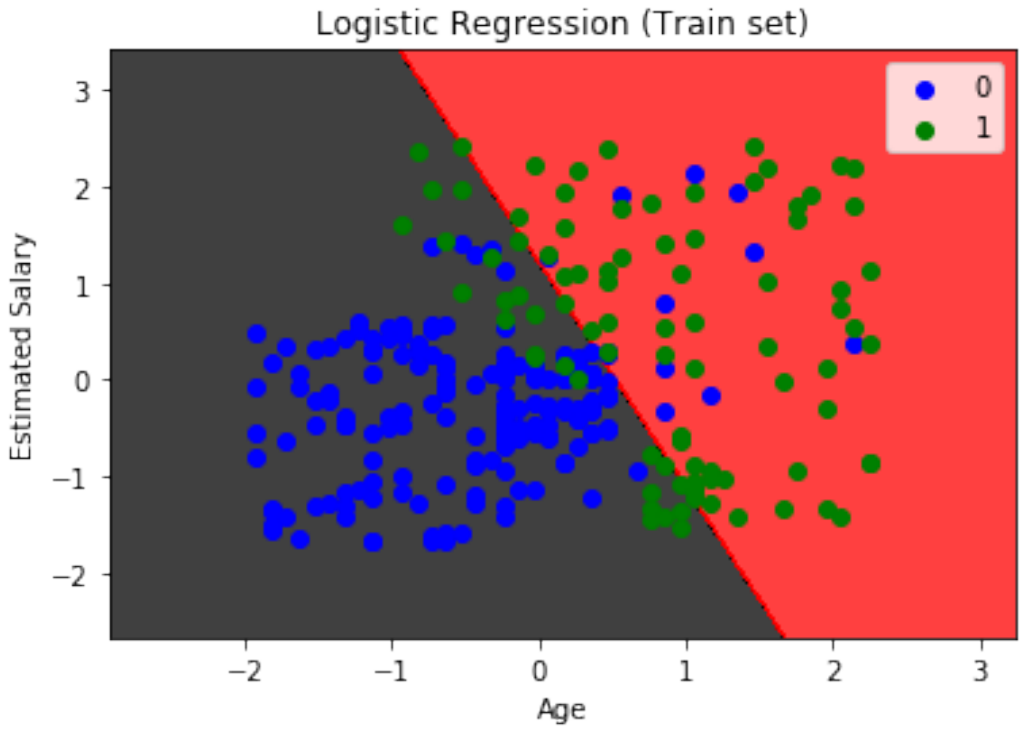
```
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0,
0,
       0, 0])

y_test

array([1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1,
0,
       1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
0,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1,
0,
       0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1,
1,
       1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0,
0,
       0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0,
0,
       0, 0])
```

Visualizing the Training Set Result

```
X_set,y_set = X_train,y_train
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,log_reg.predict(np.array([X1.ravel(),X2.ravel()]).T
).reshape(X1.shape),

alpha=0.75,cmap=ListedColormap(('black','red')))

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(['blue','green'])(i),label=j)

plt.title('Logistic Regression (Train 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 a 2-D array with a single
row if you really want 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 a 2-D array with a single
row if you really want to specify the same RGB or RGBA value for all
points.
```

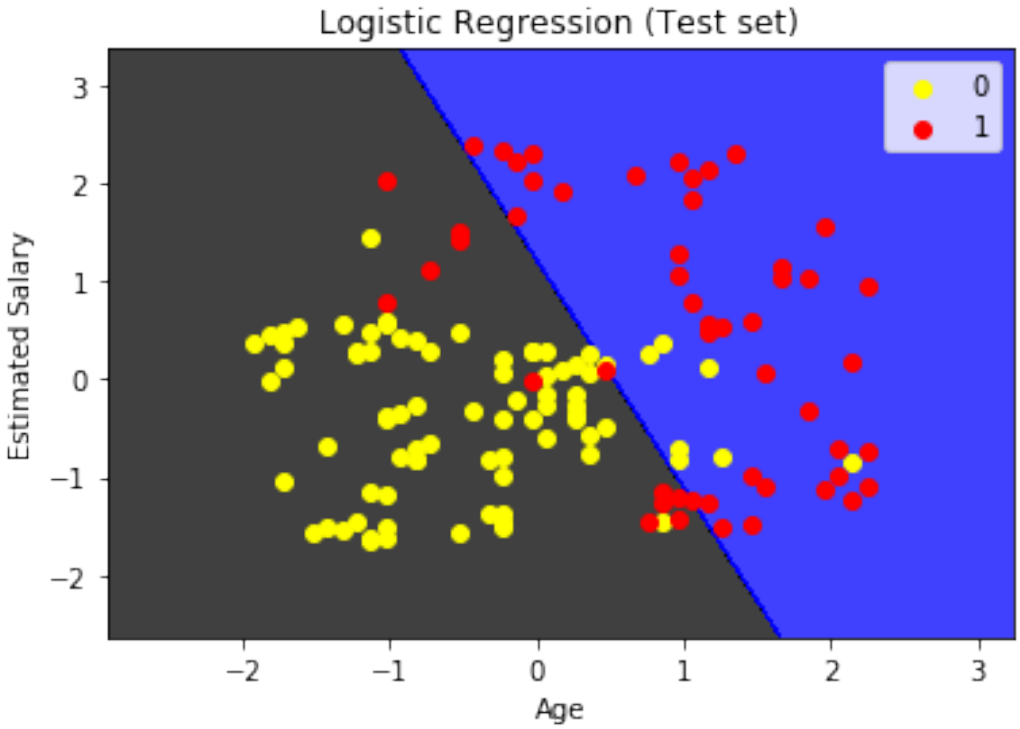


Visualizing the Testing Set results

```
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,log_reg.predict(np.array([X1.ravel(),X2.ravel()]).T
).reshape(X1.shape),

alpha=0.75,cmap=ListedColormap(('black','blue')))

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(['yellow','red'])(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 a 2-D array with a single
row if you really want 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 a 2-D array with a single
row if you really want to specify the same RGB or RGBA value for all
points.
```



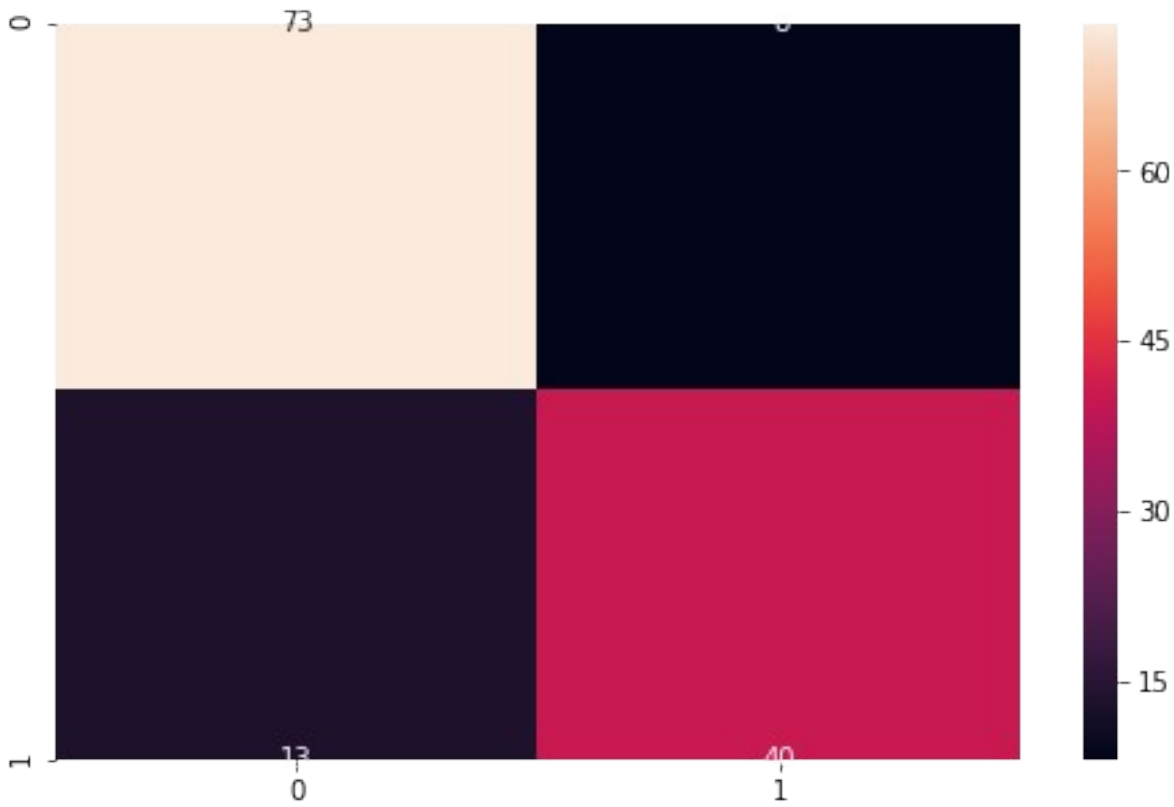
Confusion Matrix

```
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test,y_pred)
conf_matrix
```

```
array([[73,  8],
       [13, 40]])
```

```
import seaborn as sns
fig, ax = plt.subplots(figsize=(8,5))
sns.heatmap(conf_matrix, annot = True, ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff143036290>
```




```
# fig, ax = plt.subplots()
# sns.heatmap(ax=ax)
```

Accuracy

```
accuracy = (73+40)/len(y_test)
accuracy
```

```
0.8432835820895522
```

Mis Classification Rate

```
mis_cla_rate  = (13+8)/len(y_test)
mis_cla_rate
```

```
0.15671641791044777
```

Accuracy, Precision, Recall etc

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

```
Accuracy:  0.8432835820895522
```

```
print("Precision: ", metrics.precision_score(y_test,y_pred))
```

```
Precision:  0.8333333333333334
```

```
print("Recall: ", metrics.recall_score(y_test,y_pred))

Recall:  0.7547169811320755
```

ROC and AUC

```
y_pred_proba = log_reg.predict_proba(x_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.figure(figsize=(12,10))
plt.plot(fpr,tpr,label="auc="+str(auc))
plt.legend(loc=4)
plt.title("Receiver Operating Characteristic Curve (ROC)")
plt.xlabel("FPR ---->")
plt.ylabel("TPR ---->")
plt.show()
```