logistic-regression

August 25, 2023

1 Name:A.Pavani

2 Branch:Data Science

3 Roll no:20X01A6701

4 College: Narasimha Reddy Engineering College

4.1 Project Title:Prediction of "Socail_Network_Ads.csv" dataset to estimate future prediction for "age" vs "estimated salary".

##Probelm statement:A Indian news channel "zee24" has predicted salary estimation for fainancail year2018-2019. ##The organisation wants to cut off the "salary" to be safe in future by impacting huge loss.

4.2 Task:As a data science professional select the particular algorithm and predict the futurestic estimated salary.

4.3 Importing the libraries

[29]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

4.4 Importing the dataset

[30]:
```
dataset = pd.read_csv("Social_Network_Ads.csv")
dataset
```

[30]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0
..	…	…	…
395	46	41000	1

```
396   51            23000          1
397   50            20000          1
398   36            33000          0
399   49            36000          1

[400 rows x 3 columns]
```

[30]:

4.5 Splitting the dataset into the Training set and Test set

[69]:
```
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:,-1].values
from sklearn.model_selection import train_test_split
X_test,X_train,y_test,y_train = train_test_split(X,y, test_size=0.
 ↪4,random_state=0)
```

[70]:
```
print(X_train)
```

```
[[    30  87000]
 [    38  50000]
 [    35  75000]
 [    30  79000]
 [    35  50000]
 [    27  20000]
 [    31  15000]
 [    36 144000]
 [    18  68000]
 [    47  43000]
 [    30  49000]
 [    28  55000]
 [    37  55000]
 [    39  77000]
 [    20  86000]
 [    32 117000]
 [    37  77000]
 [    19  85000]
 [    55 130000]
 [    35  22000]
 [    35  47000]
 [    47 144000]
 [    41  51000]
 [    47 105000]
 [    23  28000]
 [    49 141000]
 [    28  87000]
 [    29  80000]
```

```
[    37  62000]
[    32  86000]
[    21  88000]
[    37  79000]
[    57  60000]
[    37  53000]
[    24  58000]
[    18  52000]
[    22  81000]
[    34  43000]
[    31  34000]
[    49  36000]
[    27  88000]
[    41  52000]
[    27  84000]
[    35  20000]
[    43 112000]
[    27  58000]
[    37  80000]
[    52  90000]
[    26  30000]
[    49  86000]
[    57 122000]
[    34  25000]
[    35  57000]
[    34 115000]
[    59  88000]
[    45  32000]
[    29  83000]
[    26  80000]
[    49  28000]
[    23  20000]
[    32  18000]
[    60  42000]
[    19  76000]
[    36  99000]
[    19  26000]
[    60  83000]
[    24  89000]
[    27  58000]
[    40  47000]
[    42  70000]
[    32 150000]
[    35  77000]
[    22  63000]
[    45  22000]
[    27  89000]
[    18  82000]
```

```
[    42  79000]
[    40  60000]
[    53  34000]
[    47 107000]
[    58 144000]
[    59  83000]
[    24  55000]
[    26  35000]
[    58  38000]
[    42  80000]
[    40  75000]
[    59 130000]
[    46  41000]
[    41  60000]
[    42  64000]
[    37 146000]
[    23  48000]
[    25  33000]
[    24  84000]
[    27  96000]
[    23  63000]
[    48  33000]
[    48  90000]
[    42 104000]
[    44  39000]
[    32 120000]
[    38  50000]
[    32 135000]
[    52  21000]
[    53 104000]
[    39  42000]
[    38  61000]
[    36  50000]
[    36  63000]
[    35  25000]
[    35  50000]
[    42  73000]
[    47  49000]
[    59  29000]
[    49  65000]
[    45 131000]
[    31  89000]
[    46  82000]
[    47  51000]
[    26  15000]
[    60 102000]
[    38 112000]
[    40 107000]
```

```
[    42  53000]
[    35  59000]
[    48  41000]
[    48 134000]
[    38 113000]
[    29 148000]
[    26  15000]
[    60  42000]
[    24  19000]
[    42 149000]
[    46  96000]
[    28  59000]
[    39  96000]
[    28  89000]
[    41  72000]
[    45  26000]
[    33  69000]
[    20  82000]
[    31  74000]
[    42  80000]
[    35  72000]
[    33 149000]
[    40  71000]
[    51 146000]
[    46  79000]
[    35  75000]
[    38  51000]
[    36  75000]
[    37  78000]
[    38  61000]
[    60 108000]
[    20  82000]
[    57  74000]
[    42  65000]
[    26  80000]
[    46 117000]]
```

[71]:
```
print(y_train)
```

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

[72]:
```
print(X_test)
```

```
[[    35  61000]
```

```
[    21  68000]
[    28  44000]
[    41  87000]
[    37  33000]
[    27  90000]
[    39  42000]
[    28 123000]
[    31 118000]
[    25  87000]
[    35  71000]
[    37  70000]
[    35  39000]
[    47  23000]
[    35 147000]
[    48 138000]
[    26  86000]
[    25  79000]
[    52 138000]
[    51  23000]
[    35  60000]
[    33 113000]
[    30 107000]
[    48  33000]
[    41  80000]
[    48  96000]
[    31  18000]
[    31  71000]
[    43 129000]
[    59  76000]
[    18  44000]
[    36 118000]
[    42  90000]
[    47  30000]
[    26  43000]
[    40  78000]
[    46  59000]
[    59  42000]
[    46  74000]
[    35  91000]
[    28  59000]
[    40  57000]
[    59 143000]
[    57  26000]
[    52  38000]
[    47 113000]
[    53 143000]
[    35  27000]
[    58 101000]
```

```
[    45  45000]
[    23  82000]
[    46  23000]
[    42  65000]
[    28  84000]
[    38  59000]
[    26  84000]
[    29  28000]
[    37  71000]
[    22  55000]
[    48  35000]
[    49  28000]
[    38  65000]
[    27  17000]
[    46  28000]
[    48 141000]
[    26  17000]
[    35  97000]
[    39  59000]
[    24  27000]
[    32  18000]
[    46  88000]
[    35  58000]
[    56  60000]
[    47  34000]
[    40  72000]
[    32 100000]
[    19  21000]
[    25  90000]
[    35  88000]
[    28  32000]
[    50  20000]
[    40  59000]
[    50  44000]
[    35  72000]
[    40 142000]
[    46  32000]
[    39  71000]
[    20  74000]
[    29  75000]
[    31  76000]
[    47  25000]
[    40  61000]
[    34 112000]
[    38  80000]
[    42  75000]
[    47  47000]
[    39  75000]
```

```
[    19  25000]
[    37  80000]
[    36  60000]
[    41  52000]
[    36 125000]
[    48  29000]
[    36 126000]
[    51 134000]
[    27  57000]
[    38  71000]
[    39  61000]
[    22  27000]
[    33  60000]
[    48  74000]
[    58  23000]
[    53  72000]
[    32 117000]
[    54  70000]
[    30  80000]
[    58  95000]
[    26  52000]
[    45  79000]
[    24  55000]
[    40  75000]
[    33  28000]
[    44 139000]
[    22  18000]
[    33  51000]
[    43 133000]
[    24  32000]
[    46  22000]
[    35  55000]
[    54 104000]
[    48 119000]
[    35  53000]
[    37 144000]
[    23  66000]
[    37 137000]
[    31  58000]
[    33  41000]
[    45  22000]
[    30  15000]
[    19  19000]
[    49  74000]
[    39 122000]
[    35  73000]
[    39  71000]
[    24  23000]
```

```
[    41  72000]
[    29  83000]
[    54  26000]
[    35  44000]
[    37  75000]
[    29  47000]
[    31  68000]
[    42  54000]
[    30 135000]
[    52 114000]
[    50  36000]
[    56 133000]
[    29  61000]
[    30  89000]
[    26  16000]
[    33  31000]
[    41  72000]
[    36  33000]
[    55 125000]
[    48 131000]
[    41  71000]
[    30  62000]
[    37  72000]
[    41  63000]
[    58  47000]
[    30 116000]
[    20  49000]
[    37  74000]
[    41  59000]
[    49  89000]
[    28  79000]
[    53  82000]
[    40  57000]
[    60  34000]
[    35 108000]
[    21  72000]
[    38  71000]
[    39 106000]
[    37  57000]
[    26  72000]
[    35  23000]
[    54 108000]
[    30  17000]
[    39 134000]
[    29  43000]
[    33  43000]
[    35  38000]
[    41  45000]
```

```
[    41  72000]
[    39 134000]
[    27 137000]
[    21  16000]
[    26  32000]
[    31  66000]
[    39  73000]
[    41  79000]
[    47  50000]
[    41  30000]
[    37  93000]
[    60  46000]
[    25  22000]
[    28  37000]
[    38  55000]
[    36  54000]
[    20  36000]
[    56 104000]
[    40  57000]
[    42 108000]
[    20  23000]
[    40  65000]
[    47  20000]
[    18  86000]
[    35  79000]
[    57  33000]
[    34  72000]
[    49  39000]
[    27  31000]
[    19  70000]
[    39  79000]
[    26  81000]
[    25  80000]
[    28  85000]
[    55  39000]
[    50  88000]
[    49  88000]
[    52 150000]
[    35  65000]
[    42  54000]
[    34  43000]
[    37  52000]
[    48  30000]
[    29  43000]
[    36  52000]
[    27  54000]
[    26 118000]]
```

[73]:
```
print(y_test)
```

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

4.6 Feature Scaling

[74]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

[75]:
```
print(X_train)
```

```
[[-0.69488279  0.43263217]
 [ 0.04080784 -0.66470393]
 [-0.23507615  0.07673938]
 [-0.69488279  0.19537031]
 [-0.23507615 -0.66470393]
 [-0.97076677 -1.5544359 ]
 [-0.60292146 -1.70272456]
 [-0.14311482  2.12312292]
 [-1.79841873 -0.13086474]
 [ 0.86845979 -0.87230805]
 [-0.69488279 -0.69436166]
 [-0.87880544 -0.51641526]
 [-0.05115349 -0.51641526]
 [ 0.13276917  0.13605485]
 [-1.61449607  0.40297444]
 [-0.51096013  1.32236414]
 [-0.05115349  0.13605485]
 [-1.7064574   0.37331671]
 [ 1.60415042  1.70791466]
 [-0.23507615 -1.49512043]
 [-0.23507615 -0.75367712]
 [ 0.86845979  2.12312292]
 [ 0.31669182 -0.63504619]
 [ 0.86845979  0.96647135]
 [-1.33861209 -1.31717404]
 [ 1.05238245  2.03414972]
 [-0.87880544  0.43263217]
 [-0.78684412  0.22502804]
```

```
[-0.05115349 -0.30881114]
[-0.51096013  0.40297444]
[-1.52253474  0.4622899 ]
[-0.05115349  0.19537031]
[ 1.78807308 -0.3681266 ]
[-0.05115349 -0.57573073]
[-1.24665076 -0.42744207]
[-1.79841873 -0.60538846]
[-1.43057341  0.25468578]
[-0.32703747 -0.87230805]
[-0.60292146 -1.13922764]
[ 1.05238245 -1.07991218]
[-0.97076677  0.4622899 ]
[ 0.31669182 -0.60538846]
[-0.97076677  0.34365897]
[-0.23507615 -1.5544359 ]
[ 0.50061448  1.17407548]
[-0.97076677 -0.42744207]
[-0.05115349  0.22502804]
[ 1.32826644  0.52160537]
[-1.0627281  -1.25785857]
[ 1.05238245  0.40297444]
[ 1.78807308  1.4706528 ]
[-0.32703747 -1.40614724]
[-0.23507615 -0.4570998 ]
[-0.32703747  1.26304868]
[ 1.97199573  0.4622899 ]
[ 0.68453714 -1.19854311]
[-0.78684412  0.31400124]
[-1.0627281   0.22502804]
[ 1.05238245 -1.31717404]
[-1.33861209 -1.5544359 ]
[-0.51096013 -1.61375136]
[ 2.06395706 -0.90196579]
[-1.7064574   0.10639711]
[-0.14311482  0.78852496]
[-1.7064574  -1.3764895 ]
[ 2.06395706  0.31400124]
[-1.24665076  0.49194764]
[-0.97076677 -0.42744207]
[ 0.2247305  -0.75367712]
[ 0.40865315 -0.07154928]
[-0.51096013  2.30106931]
[-0.23507615  0.13605485]
[-1.43057341 -0.27915341]
[ 0.68453714 -1.49512043]
[-0.97076677  0.49194764]
[-1.79841873  0.28434351]
```

```
[ 0.40865315  0.19537031]
[ 0.2247305  -0.3681266 ]
[ 1.42022776 -1.13922764]
[ 0.86845979  1.02578682]
[ 1.88003441  2.12312292]
[ 1.97199573  0.31400124]
[-1.24665076 -0.51641526]
[-1.0627281  -1.10956991]
[ 1.88003441 -1.02059671]
[ 0.40865315  0.22502804]
[ 0.2247305   0.07673938]
[ 1.97199573  1.70791466]
[ 0.77649847 -0.93162352]
[ 0.31669182 -0.3681266 ]
[ 0.40865315 -0.24949567]
[-0.05115349  2.18243838]
[-1.33861209 -0.72401939]
[-1.15468943 -1.16888538]
[-1.24665076  0.34365897]
[-0.97076677  0.69955176]
[-1.33861209 -0.27915341]
[ 0.96042112 -1.16888538]
[ 0.96042112  0.52160537]
[ 0.40865315  0.93681362]
[ 0.59257581 -0.99093898]
[-0.51096013  1.41133734]
[ 0.04080784 -0.66470393]
[-0.51096013  1.85620332]
[ 1.32826644 -1.52477816]
[ 1.42022776  0.93681362]
[ 0.13276917 -0.90196579]
[ 0.04080784 -0.33846887]
[-0.14311482 -0.66470393]
[-0.14311482 -0.27915341]
[-0.23507615 -1.40614724]
[-0.23507615 -0.66470393]
[ 0.40865315  0.01742392]
[ 0.86845979 -0.69436166]
[ 1.97199573 -1.28751631]
[ 1.05238245 -0.21983794]
[ 0.68453714  1.73757239]
[-0.60292146  0.49194764]
[ 0.77649847  0.28434351]
[ 0.86845979 -0.63504619]
[-1.0627281  -1.70272456]
[ 2.06395706  0.87749816]
[ 0.04080784  1.17407548]
[ 0.2247305   1.02578682]
```

```
 [ 0.40865315 -0.57573073]
 [-0.23507615 -0.39778434]
 [ 0.96042112 -0.93162352]
 [ 0.96042112  1.82654559]
 [ 0.04080784  1.20373321]
 [-0.78684412  2.24175384]
 [-1.0627281  -1.70272456]
 [ 2.06395706 -0.90196579]
 [-1.24665076 -1.58409363]
 [ 0.40865315  2.27141158]
 [ 0.77649847  0.69955176]
 [-0.87880544 -0.39778434]
 [ 0.13276917  0.69955176]
 [-0.87880544  0.49194764]
 [ 0.31669182 -0.01223381]
 [ 0.68453714 -1.3764895 ]
 [-0.4189988  -0.10120701]
 [-1.61449607  0.28434351]
 [-0.60292146  0.04708165]
 [ 0.40865315  0.22502804]
 [-0.23507615 -0.01223381]
 [-0.4189988   2.27141158]
 [ 0.2247305  -0.04189155]
 [ 1.23630511  2.18243838]
 [ 0.77649847  0.19537031]
 [-0.23507615  0.07673938]
 [ 0.04080784 -0.63504619]
 [-0.14311482  0.07673938]
 [-0.05115349  0.16571258]
 [ 0.04080784 -0.33846887]
 [ 2.06395706  1.05544455]
 [-1.61449607  0.28434351]
 [ 1.78807308  0.04708165]
 [ 0.40865315 -0.21983794]
 [-1.0627281   0.22502804]
 [ 0.77649847  1.32236414]]
```

[76]:
```
print(X_test)
```

```
[[-0.23507615 -0.33846887]
 [-1.52253474 -0.13086474]
 [-0.87880544 -0.84265032]
 [ 0.31669182  0.43263217]
 [-0.05115349 -1.16888538]
 [-0.97076677  0.52160537]
 [ 0.13276917 -0.90196579]
 [-0.87880544  1.50031054]
 [-0.60292146  1.35202187]
```

```
[-1.15468943  0.43263217]
[-0.23507615 -0.04189155]
[-0.05115349 -0.07154928]
[-0.23507615 -0.99093898]
[ 0.86845979 -1.4654627 ]
[-0.23507615  2.21209611]
[ 0.96042112  1.94517652]
[-1.0627281   0.40297444]
[-1.15468943  0.19537031]
[ 1.32826644  1.94517652]
[ 1.23630511 -1.4654627 ]
[-0.23507615 -0.3681266 ]
[-0.4189988   1.20373321]
[-0.69488279  1.02578682]
[ 0.96042112 -1.16888538]
[ 0.31669182  0.22502804]
[ 0.96042112  0.69955176]
[-0.60292146 -1.61375136]
[-0.60292146 -0.04189155]
[ 0.50061448  1.67825693]
[ 1.97199573  0.10639711]
[-1.79841873 -0.84265032]
[-0.14311482  1.35202187]
[ 0.40865315  0.52160537]
[ 0.86845979 -1.25785857]
[-1.0627281  -0.87230805]
[ 0.2247305   0.16571258]
[ 0.77649847 -0.39778434]
[ 1.97199573 -0.90196579]
[ 0.77649847  0.04708165]
[-0.23507615  0.5512631 ]
[-0.87880544 -0.39778434]
[ 0.2247305  -0.4570998 ]
[ 1.97199573  2.09346518]
[ 1.78807308 -1.3764895 ]
[ 1.32826644 -1.02059671]
[ 0.86845979  1.20373321]
[ 1.42022776  2.09346518]
[-0.23507615 -1.34683177]
[ 1.88003441  0.84784042]
[ 0.68453714 -0.81299259]
[-1.33861209  0.28434351]
[ 0.77649847 -1.4654627 ]
[ 0.40865315 -0.21983794]
[-0.87880544  0.34365897]
[ 0.04080784 -0.39778434]
[-1.0627281   0.34365897]
[-0.78684412 -1.31717404]
```

```
[-0.05115349 -0.04189155]
[-1.43057341 -0.51641526]
[ 0.96042112 -1.10956991]
[ 1.05238245 -1.31717404]
[ 0.04080784 -0.21983794]
[-0.97076677 -1.64340909]
[ 0.77649847 -1.31717404]
[ 0.96042112  2.03414972]
[-1.0627281  -1.64340909]
[-0.23507615  0.72920949]
[ 0.13276917 -0.39778434]
[-1.24665076 -1.34683177]
[-0.51096013 -1.61375136]
[ 0.77649847  0.4622899 ]
[-0.23507615 -0.42744207]
[ 1.69611175 -0.3681266 ]
[ 0.86845979 -1.13922764]
[ 0.2247305  -0.01223381]
[-0.51096013  0.81818269]
[-1.7064574  -1.52477816]
[-1.15468943  0.52160537]
[-0.23507615  0.4622899 ]
[-0.87880544 -1.19854311]
[ 1.14434378 -1.5544359 ]
[ 0.2247305  -0.39778434]
[ 1.14434378 -0.84265032]
[-0.23507615 -0.01223381]
[ 0.2247305   2.06380745]
[ 0.77649847 -1.19854311]
[ 0.13276917 -0.04189155]
[-1.61449607  0.04708165]
[-0.78684412  0.07673938]
[-0.60292146  0.10639711]
[ 0.86845979 -1.40614724]
[ 0.2247305  -0.33846887]
[-0.32703747  1.17407548]
[ 0.04080784  0.22502804]
[ 0.40865315  0.07673938]
[ 0.86845979 -0.75367712]
[ 0.13276917  0.07673938]
[-1.7064574  -1.40614724]
[-0.05115349  0.22502804]
[-0.14311482 -0.3681266 ]
[ 0.31669182 -0.60538846]
[-0.14311482  1.559626  ]
[ 0.96042112 -1.28751631]
[-0.14311482  1.58928373]
[ 1.23630511  1.82654559]
```

```
[-0.97076677 -0.4570998 ]
[ 0.04080784 -0.04189155]
[ 0.13276917 -0.33846887]
[-1.43057341 -1.34683177]
[-0.4189988  -0.3681266 ]
[ 0.96042112  0.04708165]
[ 1.88003441 -1.4654627 ]
[ 1.42022776 -0.01223381]
[-0.51096013  1.32236414]
[ 1.51218909 -0.07154928]
[-0.69488279  0.22502804]
[ 1.88003441  0.66989403]
[-1.0627281  -0.60538846]
[ 0.68453714  0.19537031]
[-1.24665076 -0.51641526]
[ 0.2247305   0.07673938]
[-0.4189988  -1.31717404]
[ 0.59257581  1.97483425]
[-1.43057341 -1.61375136]
[-0.4189988  -0.63504619]
[ 0.50061448  1.79688786]
[-1.24665076 -1.19854311]
[ 0.77649847 -1.49512043]
[-0.23507615 -0.51641526]
[ 1.51218909  0.93681362]
[ 0.96042112  1.38167961]
[-0.23507615 -0.57573073]
[-0.05115349  2.12312292]
[-1.33861209 -0.19018021]
[-0.05115349  1.91551879]
[-0.60292146 -0.42744207]
[-0.4189988  -0.93162352]
[ 0.68453714 -1.49512043]
[-0.69488279 -1.70272456]
[-1.7064574  -1.58409363]
[ 1.05238245  0.04708165]
[ 0.13276917  1.4706528 ]
[-0.23507615  0.01742392]
[ 0.13276917 -0.04189155]
[-1.24665076 -1.4654627 ]
[ 0.31669182 -0.01223381]
[-0.78684412  0.31400124]
[ 1.51218909 -1.3764895 ]
[-0.23507615 -0.84265032]
[-0.05115349  0.07673938]
[-0.78684412 -0.75367712]
[-0.60292146 -0.13086474]
[ 0.40865315 -0.546073  ]
```

```
[-0.69488279  1.85620332]
[ 1.32826644  1.23339094]
[ 1.14434378 -1.07991218]
[ 1.69611175  1.79688786]
[-0.78684412 -0.33846887]
[-0.69488279  0.49194764]
[-1.0627281  -1.67306683]
[-0.4189988  -1.22820084]
[ 0.31669182 -0.01223381]
[-0.14311482 -1.16888538]
[ 1.60415042  1.559626  ]
[ 0.96042112  1.73757239]
[ 0.31669182 -0.04189155]
[-0.69488279 -0.30881114]
[-0.05115349 -0.01223381]
[ 0.31669182 -0.27915341]
[ 1.88003441 -0.75367712]
[-0.69488279  1.29270641]
[-1.61449607 -0.69436166]
[-0.05115349  0.04708165]
[ 0.31669182 -0.39778434]
[ 1.05238245  0.49194764]
[-0.87880544  0.19537031]
[ 1.42022776  0.28434351]
[ 0.2247305  -0.4570998 ]
[ 2.06395706 -1.13922764]
[-0.23507615  1.05544455]
[-1.52253474 -0.01223381]
[ 0.04080784 -0.04189155]
[ 0.13276917  0.99612909]
[-0.05115349 -0.4570998 ]
[-1.0627281  -0.01223381]
[-0.23507615 -1.4654627 ]
[ 1.51218909  1.05544455]
[-0.69488279 -1.64340909]
[ 0.13276917  1.82654559]
[-0.78684412 -0.87230805]
[-0.4189988  -0.87230805]
[-0.23507615 -1.02059671]
[ 0.31669182 -0.81299259]
[ 0.31669182 -0.01223381]
[ 0.13276917  1.82654559]
[-0.97076677  1.91551879]
[-1.52253474 -1.67306683]
[-1.0627281  -1.19854311]
[-0.60292146 -0.19018021]
[ 0.13276917  0.01742392]
[ 0.31669182  0.19537031]
```

```
[ 0.86845979 -0.66470393]
[ 0.31669182 -1.25785857]
[-0.05115349  0.61057856]
[ 2.06395706 -0.78333486]
[-1.15468943 -1.49512043]
[-0.87880544 -1.05025445]
[ 0.04080784 -0.51641526]
[-0.14311482 -0.546073  ]
[-1.61449607 -1.07991218]
[ 1.69611175  0.93681362]
[ 0.2247305  -0.4570998 ]
[ 0.40865315  1.05544455]
[-1.61449607 -1.4654627 ]
[ 0.2247305  -0.21983794]
[ 0.86845979 -1.5544359 ]
[-1.79841873  0.40297444]
[-0.23507615  0.19537031]
[ 1.78807308 -1.16888538]
[-0.32703747 -0.01223381]
[ 1.05238245 -0.99093898]
[-0.97076677 -1.22820084]
[-1.7064574  -0.07154928]
[ 0.13276917  0.19537031]
[-1.0627281   0.25468578]
[-1.15468943  0.22502804]
[-0.87880544  0.37331671]
[ 1.60415042 -0.99093898]
[ 1.14434378  0.4622899 ]
[ 1.05238245  0.4622899 ]
[ 1.32826644  2.30106931]
[-0.23507615 -0.21983794]
[ 0.40865315 -0.546073  ]
[-0.32703747 -0.87230805]
[-0.05115349 -0.60538846]
[ 0.96042112 -1.25785857]
[-0.78684412 -0.87230805]
[-0.14311482 -0.60538846]
[-0.97076677 -0.546073  ]
[-1.0627281   1.35202187]]
```

4.7 Training the Logistic Regression model on the Training set

[77]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

[77]:
```
LogisticRegression(random_state=0)
```

4.8 Predicting a new result

[78]:
```
print(classifier.predict(sc.transform([[47,43000]])))
```

```
[0]
```

4.9 Predicting the Test set results

[79]:
```
y_pred = classifier.predict(X_test)
x=print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.
 ↪reshape(len(y_test),1)),1))
```

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

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

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

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

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

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

4.10 Making the Confusion Matrix

[80]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

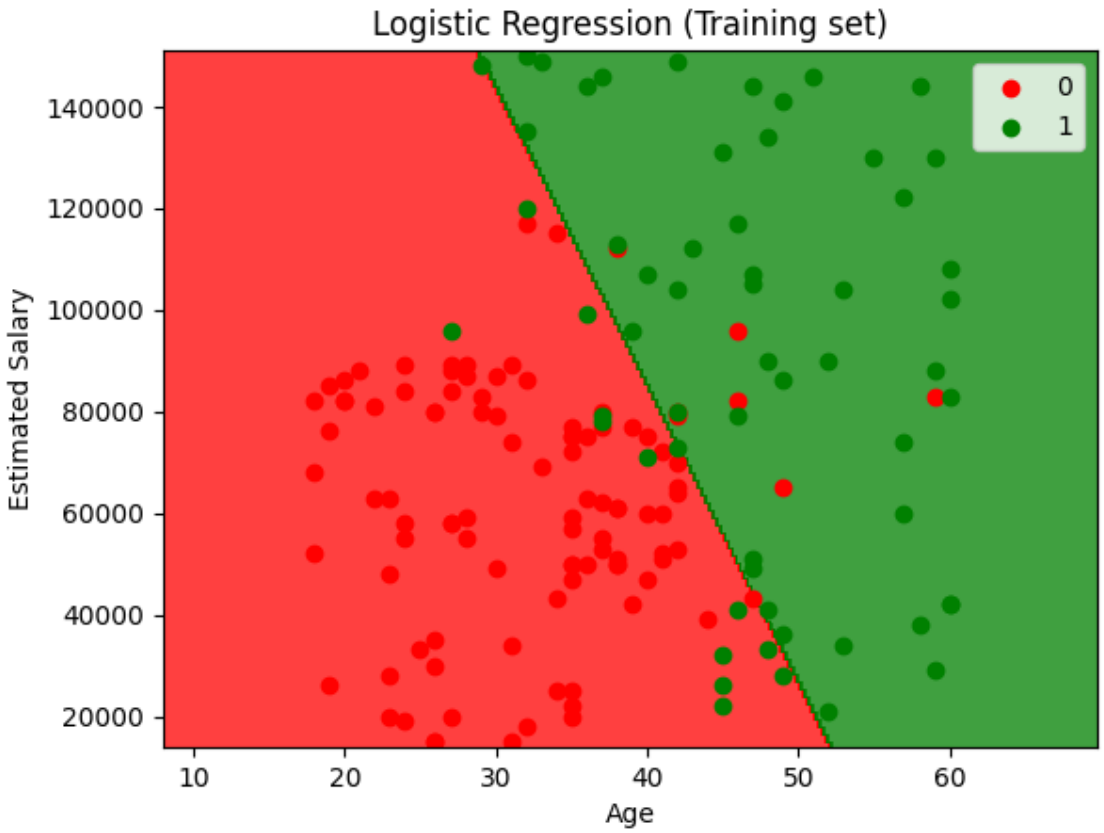
```
[[143  13]
 [ 29  55]]
```

[80]:
```
0.825
```

4.11 Visualising the Training set results

[81]:
```
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 (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
<ipython-input-81-3277c112bab0>:10: 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], c =
ListedColormap(('red', 'green'))(i), label = j)
```

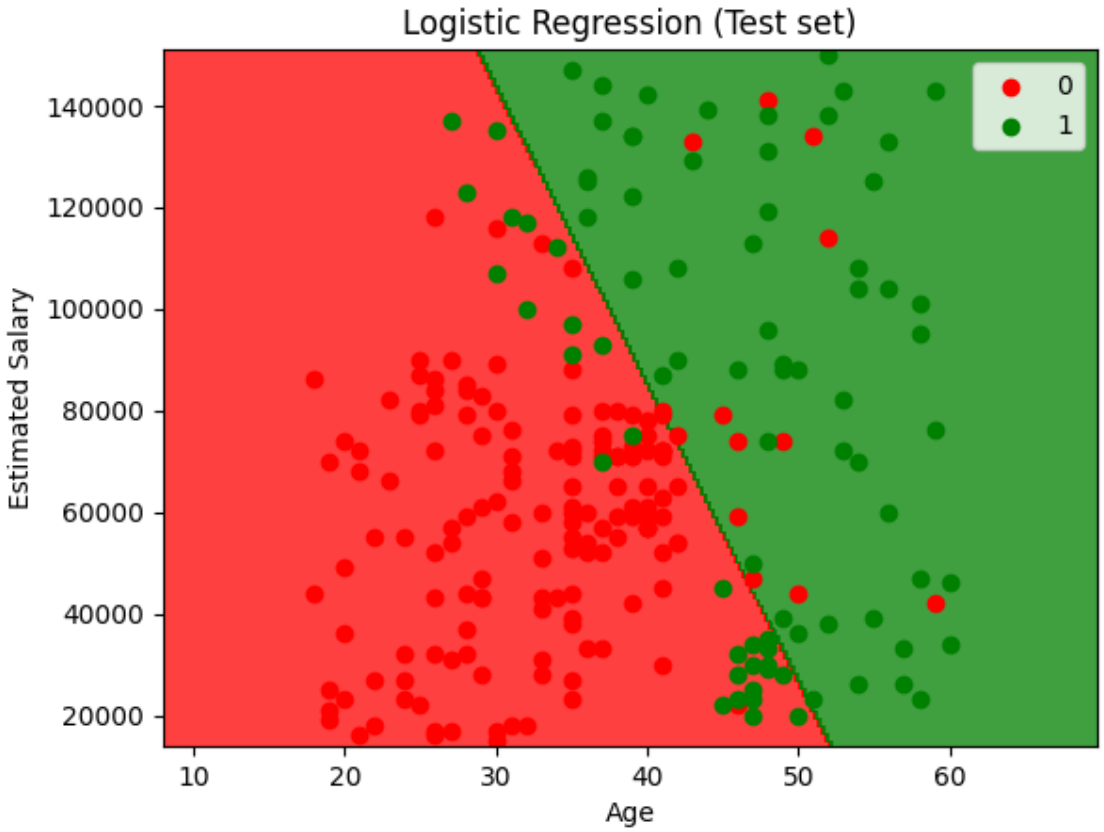


4.12 Visualising the Test set results

[82]:
```
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()
```

```
<ipython-input-82-53d83417cfe6>:10: 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], c =
ListedColormap(('red', 'green'))(i), label = j)
```



[ ]: