

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

```
In [4]: dataset = pd.read_csv('Social_Network_Ads (1).csv')
```

```
In [5]: dataset
```

```
Out[5]:
```

|     | 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 × 5 columns

```
In [6]: dataset.head()
```

Out [6]:

|   | 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         |

In [7]: `dataset.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
```

In [8]: `X = dataset.iloc[:, [2, 3]].values`  
`y = dataset.iloc[:, 4].values`

```
print(X[:3, :])
print('-'*15)
print(y[:3])
```

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

```
-----
[0 0 0]
```

```
In [9]: dataset.tail()
```

```
Out[9]:
```

|     | 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         |

```
In [10]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

```
print(X_train[:3])
print('-'*15)
print(y_train[:3])
print('-'*15)
print(X_test[:3])
print('-'*15)
print(y_test[:3])
```

```
[[ 44 39000]
 [ 32 120000]
 [ 38 50000]]
```

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

```
-----
[[ 30 87000]
 [ 38 50000]
 [ 35 75000]]
```

```
-----
[0 0 0]
```

```
In [11]: from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
```

```
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

```
In [12]: print(X_train[:3])
print('-'*15)
print(X_test[:3])
```

```
[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824  ]]
```

-----

```
[[ -0.80480212  0.50496393]
 [-0.01254409 -0.5677824  ]
 [-0.30964085  0.1570462  ]]
```

```
In [13]: from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0, solver='lbfgs' )
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

print(X_test[:10])
print('-'*15)
print(y_pred[:10])
```

```
[[ -0.80480212  0.50496393]
 [-0.01254409 -0.5677824  ]
 [-0.30964085  0.1570462  ]
 [-0.80480212  0.27301877]
 [-0.30964085 -0.5677824  ]
 [-1.10189888 -1.43757673]
 [-0.70576986 -1.58254245]
 [-0.21060859  2.15757314]
 [-1.99318916 -0.04590581]
 [ 0.8787462  -0.77073441]]
```

-----

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

```
In [14]: print(y_pred[:20])
print(y_test[:20])
```

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

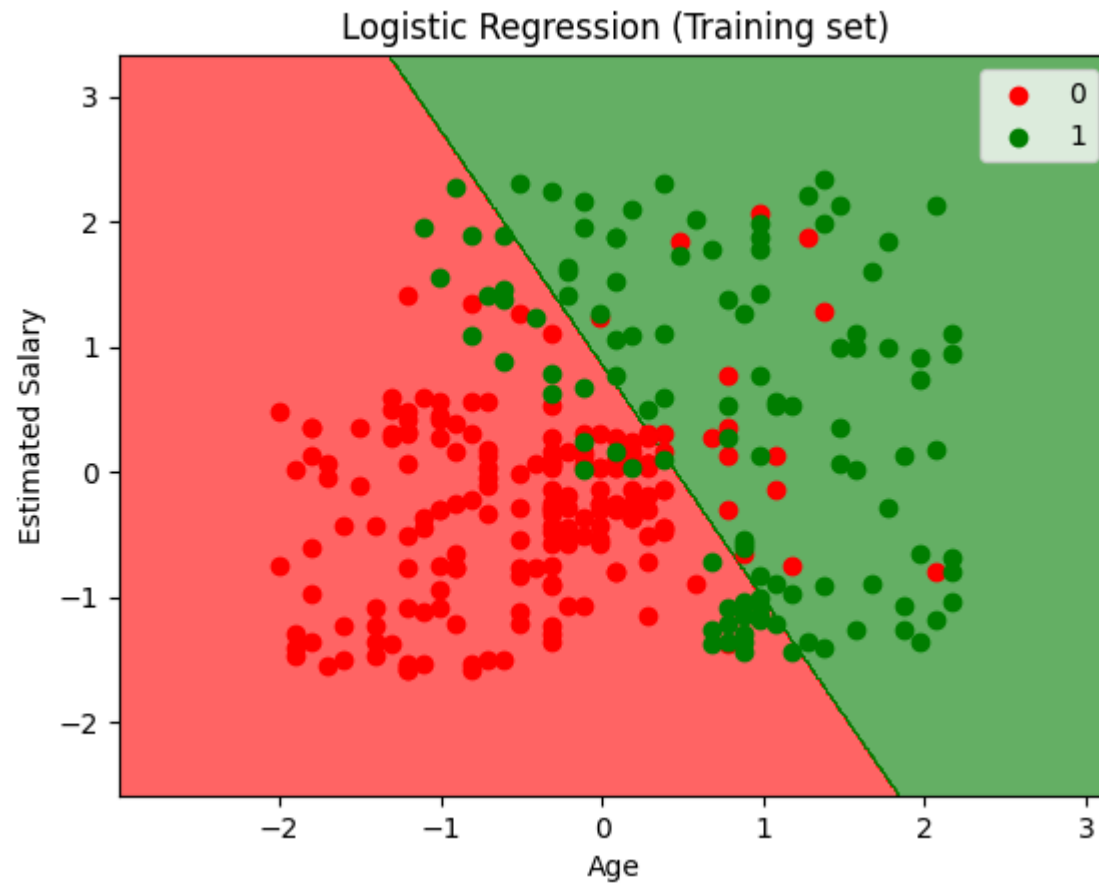
```
In [15]: from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

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

```
In [16]: from matplotlib.colors import ListedColormap
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, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.6, 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()
```

/tmp/ipykernel\_1856/2759427564.py: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],
```

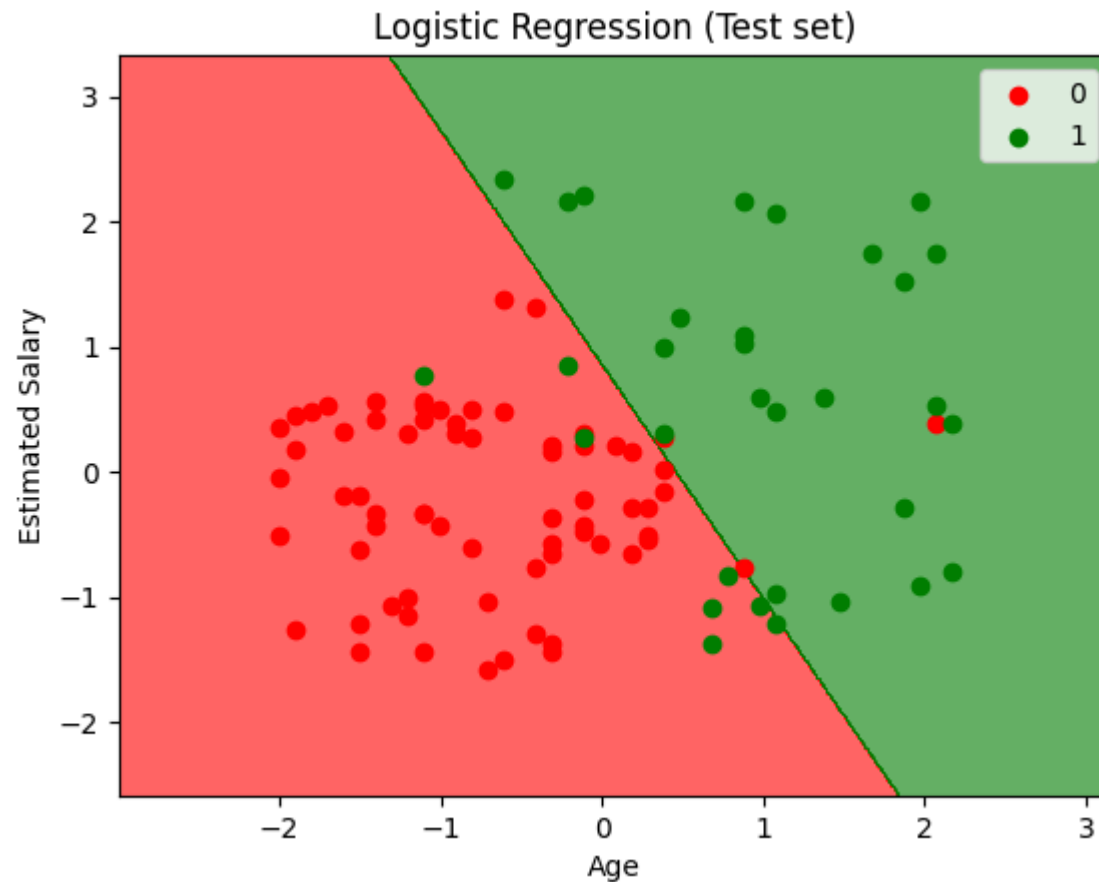


```
In [17]: from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                    np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.6, 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()
```

/tmp/ipykernel\_1856/238990405.py: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],
```



In [ ]: