In [ ]:
```
#Objective
# 1. implement logistic regression using python to perform classification on Social_Ne
# 2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision,
```

In [5]:
```
pip install sklearn
```

```
Collecting sklearn
  Downloading sklearn-0.0.tar.gz (1.1 kB)
Note: you may need to restart the kernel to use updated packages.
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting scikit-learn
  Downloading scikit_learn-1.0.2-cp39-cp39-win_amd64.whl (7.2 MB)
     ---------------------------------------- 7.2/7.2 MB 1.4 MB/s eta 0:00:00
Requirement already satisfied: scipy>=1.1.0 in c:\users\rishu\appdata\local\programs
\python\python39\lib\site-packages (from scikit-learn->sklearn) (1.8.0)
Requirement already satisfied: joblib>=0.11 in c:\users\rishu\appdata\local\programs
\python\python39\lib\site-packages (from scikit-learn->sklearn) (1.1.0)
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-3.1.0-py3-none-any.whl (14 kB)
Requirement already satisfied: numpy>=1.14.6 in c:\users\rishu\appdata\local\programs
\python\python39\lib\site-packages (from scikit-learn->sklearn) (1.21.3)
Using legacy 'setup.py install' for sklearn, since package 'wheel' is not installed.
Installing collected packages: threadpoolctl, scikit-learn, sklearn
  Running setup.py install for sklearn: started
  Running setup.py install for sklearn: finished with status 'done'
Successfully installed scikit-learn-1.0.2 sklearn-0.0 threadpoolctl-3.1.0
```

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('social_network_ads.csv')
dataset.head(20)
```

Out[2]:

	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
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0
13	15704987	Male	32	18000	0
14	15628972	Male	18	82000	0
15	15697686	Male	29	80000	0
16	15733883	Male	47	25000	1
17	15617482	Male	45	26000	1
18	15704583	Male	46	28000	1
19	15621083	Female	48	29000	1

In [3]:
```
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]:
```
# split our data into two sets: a training set for the machine to learn from,
# as well as a test set for the machine to execute on

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_sta
```

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]:
```
 from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0, solver='lbfgs' )

# fit the classifier to the training set with the aptly named .fit() method so that it
classifier.fit(X_train, y_train)

# predict() method will give us a vector of predictions for our dataset, X_test.
y_pred = classifier.predict(X_test)


# we will test the classifier's predictive power on the test set.
print(X_test[:10])

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

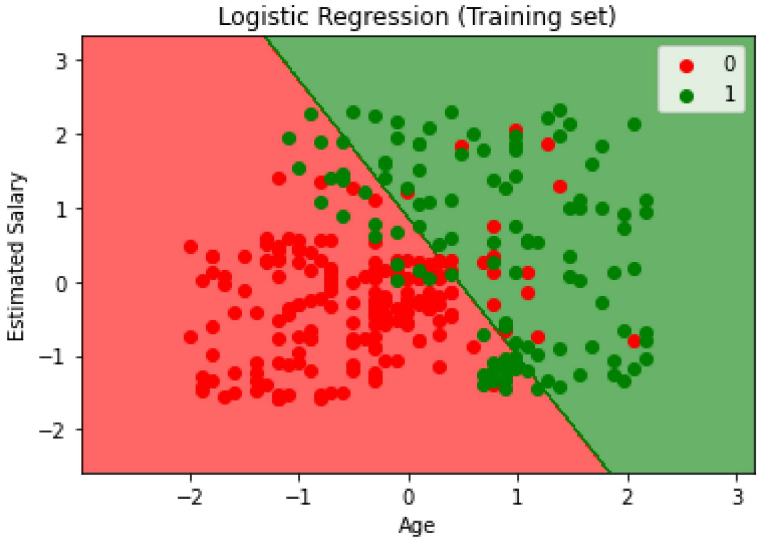
```
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]
 [-0.80480212  0.27301877]
 [-0.30964085 -0.5677824 ]
 [-1.10189888 -1.43757673]
 [-0.70576986 -1.58254245]
 [-0.21060859  2.15757314]
 [-1.99318916 -0.04590581]
 [ 0.8787462  -0.77073441]]
---------------
[0 0 0 0 0 0 0 1 0 1]
```

In [13]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

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

In [14]:
```
# Visualizing the Training set results
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()
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max()
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(
             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()
```

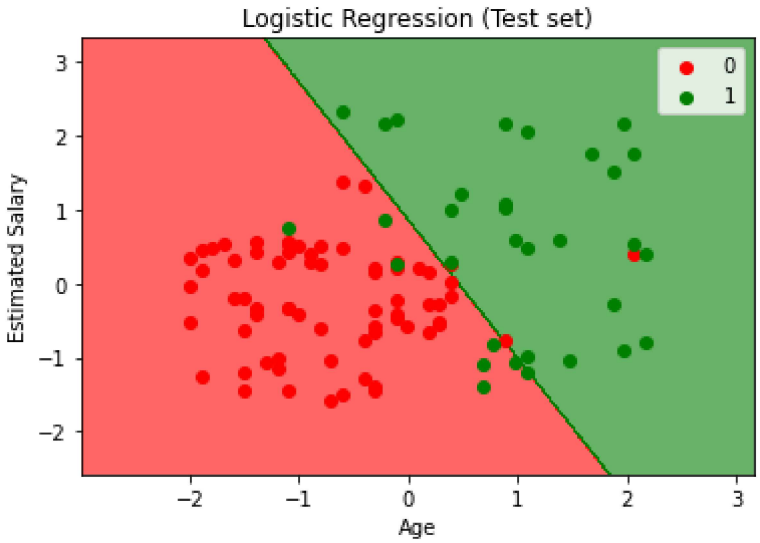
```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoide
d as value-mapping will have precedence in case its length matches with *x* & *y*.  P
lease use the *color* keyword-argument or provide a 2D array with a single row if you
intend to specify the same RGB or RGBA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



In [15]:

```
# Visualizing the Test set results
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()
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max()
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(
             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()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoide
d as value-mapping will have precedence in case its length matches with *x* & *y*.  P
lease 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.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoide
d as value-mapping will have precedence in case its length matches with *x* & *y*.  P
lease 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.
```


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

In [ ]: