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

In [3]:
```
dataset = pd.read_csv('Social_Network_Ads.csv')
```

In [4]:
```
print(dataset.columns)
```

```
Index(['Age', 'EstimatedSalary', 'Purchased'], dtype='object')
```

In [5]:
```
X = dataset.iloc[:, [0, 1]].values
y = dataset.iloc[:, 2].values
```

In [6]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
```

In [8]:
```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [9]:
```
# Fitting Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(random_state = 0)
log_reg.fit(X_train, y_train)
```

Out[9]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_interc
ept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=10
0,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verb
ose=0,
                   warm_start=False)
```

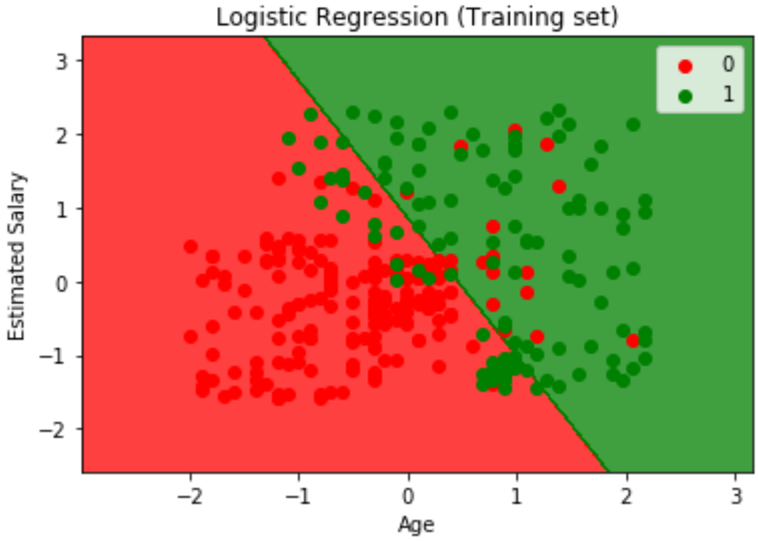
In [12]:
```
# Predicting the Test set results
y_pred = log_reg.predict(X_test)
```

In [13]:
```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

In [14]:

```
# Visualising 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 =
                     np.arange(start = X_set[:, 1].min() - 1, stop =
plt.contourf(X1, X2, log_reg.predict(np.array([X1.ravel(), X2.ravel(
             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()
```

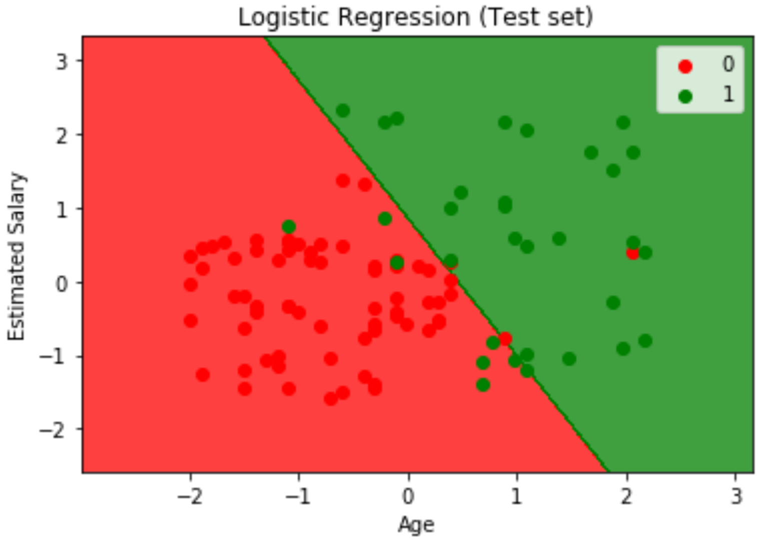
```
'c' argument looks like a single numeric RGB or RGBA sequence, whic
h should be avoided as value-mapping will have precedence in case i
ts length matches with 'x' & 'y'.  Please use a 2-D array with a si
ngle row if you really want to specify the same RGB or RGBA value f
or all points.
'c' argument looks like a single numeric RGB or RGBA sequence, whic
h should be avoided as value-mapping will have precedence in case i
ts length matches with 'x' & 'y'.  Please use a 2-D array with a si
ngle row if you really want to specify the same RGB or RGBA value f
or all points.
```



In [15]:

```
# Visualising 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 =
                     np.arange(start = X_set[:, 1].min() - 1, stop =
plt.contourf(X1, X2, log_reg.predict(np.array([X1.ravel(), X2.ravel()
             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()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, whic
h should be avoided as value-mapping will have precedence in case i
ts length matches with 'x' & 'y'.  Please use a 2-D array with a si
ngle row if you really want to specify the same RGB or RGBA value f
or all points.
'c' argument looks like a single numeric RGB or RGBA sequence, whic
h should be avoided as value-mapping will have precedence in case i
ts length matches with 'x' & 'y'.  Please use a 2-D array with a si
ngle row if you really want to specify the same RGB or RGBA value f
or all points.
```



In [16]:

```
# Logistic Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [17]:

```
X = dataset.iloc[:, [0, 1]].values
y = dataset.iloc[:, 2].values
```

In [18]:

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
```

In [19]:

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [20]:

```
# Fitting Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(random_state = 0)
log_reg.fit(X_train, y_train)
```

Out[20]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_interc
ept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=10
0,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verb
ose=0,
                   warm_start=False)
```

In [21]:

```
# Predicting the Test set results
y_pred = log_reg.predict(X_test)
```

In [22]:

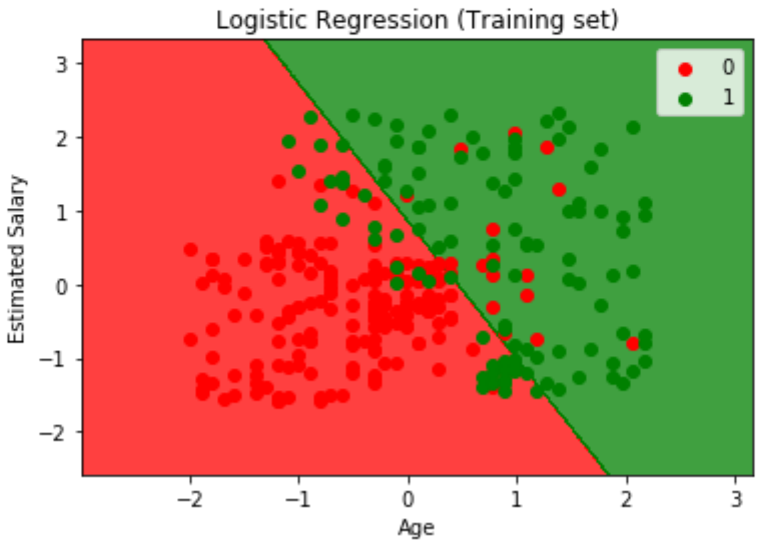
```
# Making the Confusion Matrix

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

Out[22]:

```
array([[65,  3],
       [ 8, 24]])
```

In [23]:
```
# Visualising 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 =
                     np.arange(start = X_set[:, 1].min() - 1, stop =
plt.contourf(X1, X2, log_reg.predict(np.array([X1.ravel(), X2.ravel(
             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()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, whic
h should be avoided as value-mapping will have precedence in case i
ts length matches with 'x' & 'y'.  Please use a 2-D array with a si
ngle row if you really want to specify the same RGB or RGBA value f
or all points.
'c' argument looks like a single numeric RGB or RGBA sequence, whic
h should be avoided as value-mapping will have precedence in case i
ts length matches with 'x' & 'y'.  Please use a 2-D array with a si
ngle row if you really want to specify the same RGB or RGBA value f
or all points.
```



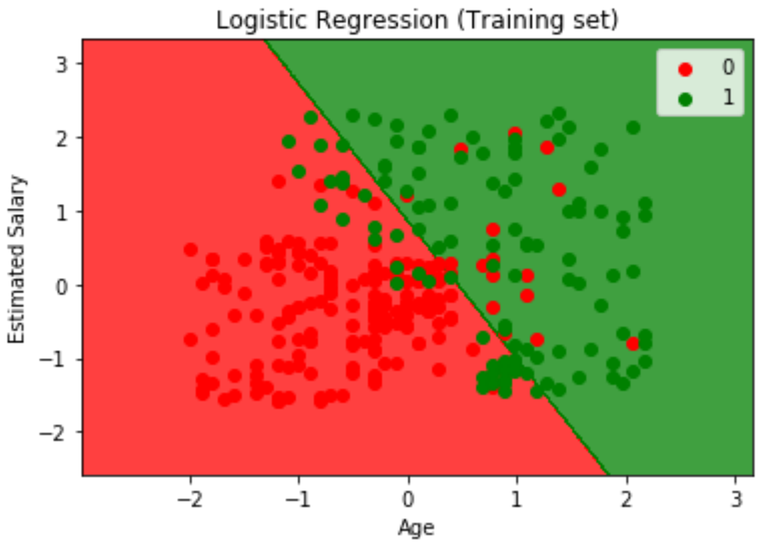
In [ ]:

In [ ]:

In [24]:

```
# Visualising 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 =
                     np.arange(start = X_set[:, 1].min() - 1, stop =
plt.contourf(X1, X2, log_reg.predict(np.array([X1.ravel(), X2.ravel(
             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()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, whic
h should be avoided as value-mapping will have precedence in case i
ts length matches with 'x' & 'y'.  Please use a 2-D array with a si
ngle row if you really want to specify the same RGB or RGBA value f
or all points.
'c' argument looks like a single numeric RGB or RGBA sequence, whic
h should be avoided as value-mapping will have precedence in case i
ts length matches with 'x' & 'y'.  Please use a 2-D array with a si
ngle row if you really want to specify the same RGB or RGBA value f
or all points.
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: