In [186]:
```
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 sklearn.metrics import confusion_matrix
```

In [168]:
```
data=pd.read_csv("Social_Network_Ads.csv")
data.head(10)
```

Out[168]:

	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

In [169]:
```
real_x=data.iloc[:,2:4].values
real_y=data.iloc[:,4].values
```

In [170]:
```
training_x,test_x,training_y,test_y=train_test_split(real_x,real_y,test_size=0.25,random_state=0)
```

In [171]:
```
scaler= StandardScaler()
training_x = scaler.fit_transform(training_x)
test_x=scaler.fit_transform(test_x)
```

In [172]:
```
reg_classifer=LogisticRegression(random_state=0)
reg_classifer.fit(training_x,training_y)
```

Out[172]:
```
LogisticRegression(random_state=0)
```

In [173]:
```
y_pred=reg_classifer.predict(test_x)
y_pred
```

Out[173]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1], dtype=int64)
```

In [174]:
```
test_y
```

Out[174]:
```
array([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], dtype=int64)
```

In [175]:
```
c_m=confusion_matrix(test_y,y_pred)
c_m
```

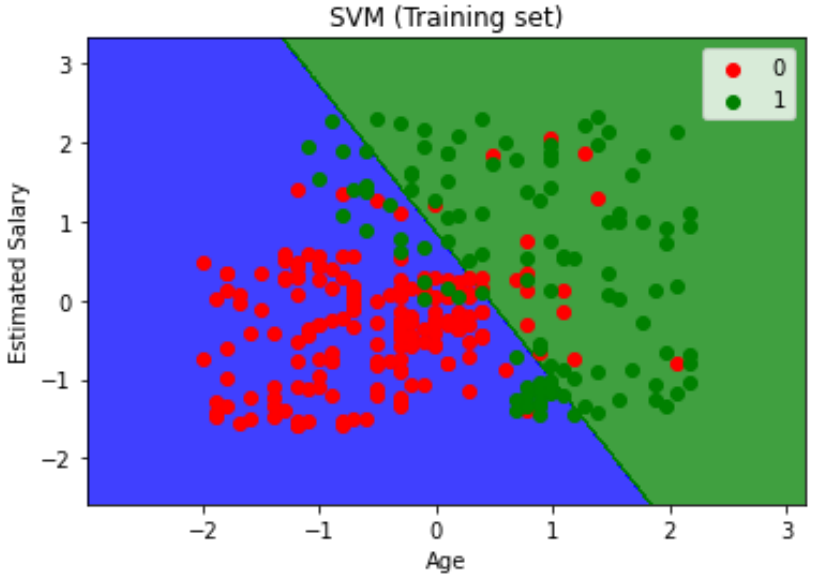
Out[175]:
```
array([[63,  5],
       [ 8, 24]], dtype=int64)
```

In [ ]:

In [176]:
```
# # import matplotlib.pyplot as plt
# from matplotlib.colors import ListedColormap

# x_set,y_set =training_x,training_y
# x1, x2 =np.meshgrid(np.arange(start=x_set[:,0].min() -1,stop =x_set[:,0] +1,step=0.01  ),
#                     np.arange(start=x_set[:,1].min() -1,stop =x_set[:,1] +1,step=0.01  )


# plt.contourf(x1,x2,reg_classifer.predict(np.arange([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)):
#                     c=ListedColormap(("red","green"))(i),label=j)

# plt.title("logical regession (traning set)")
# plt.xlabel("Age")
# plt.ylabel("Estimated salary")
# plt.legend()
# plt.show()
```

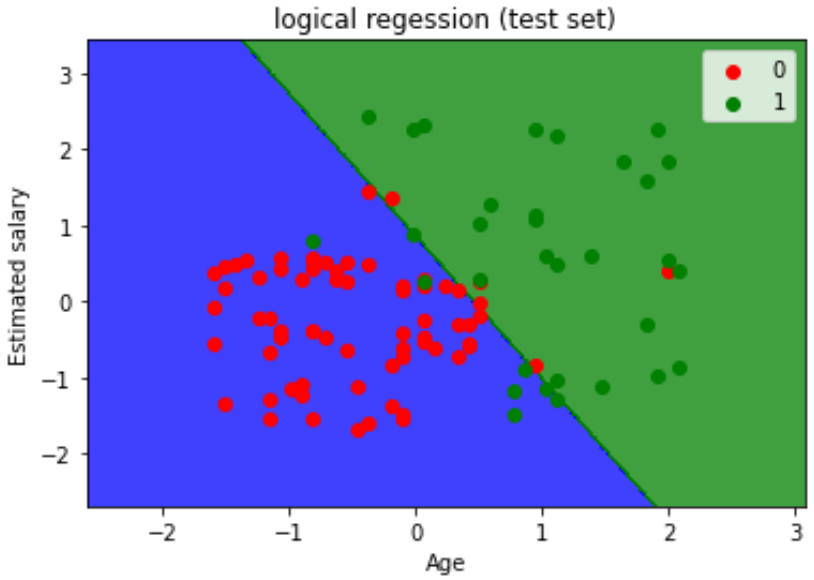
In [194]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = training_x,training_y
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, reg_classifer.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('blue', '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('logical regession  (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 avoided as value-mapping wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly 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 wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly want to specify the same RGB or RGBA value for all points.
```


SVM (Training set)
Estimated Salary
Age
0
1
3
2
1
0
−1
−2
−2
−1
0
1
2
3

In [182]:
```
from matplotlib.colors import ListedColormap
X_set,Y_set =test_x,test_y
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, reg_classifer.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('blue', '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("logical regession (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 wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly 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 wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly want to specify the same RGB or RGBA value for all points.
```


logical regession (test set)
Estimated salary
Age
0
1
3
2
1
0
−1
−2
−2
−1
0
1
2
3