In [7]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [8]:
```
dataset = pd.read_csv('Social_Network_Ads.csv')
```

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

Out[9]:

	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 [10]:
```
dataset.head(20)
```

Out[10]:

	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

	User ID	Gender	Age	EstimatedSalary	Purchased
18	15704583	Male	46	28000	1
19	15621083	Female	48	29000	1

In [11]:
```
#In our Data set we'll consider Age and  EstimatedSalary as Independent variable (X1 an
X = dataset.iloc[:, [2,3]].values
Y = dataset.iloc[:, 4].values
#see here: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.
```

In [12]:
```
X[0:10]
```

Out[12]:
```
array([[    19,  19000],
       [    35,  20000],
       [    26,  43000],
       [    27,  57000],
       [    19,  76000],
       [    27,  58000],
       [    27,  84000],
       [    32, 150000],
       [    25,  33000],
       [    35,  65000]], dtype=int64)
```

In [14]:
```
#Now we'll split our Data set into Training Data and Test Data. Training data will be u
#Logistic model and Test data will be used to validate our model. We'll use Sklearn to
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_stat
```

In [15]:
```
#Now we'll do feature scaling to scale our data between 0 and 1 to get better accuracy.
#Here Scaling is important because there is a huge difference between Age and Estimated

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

In [16]:
```
#Import LogisticRegression from sklearn.linear_model
#Make an instance classifier of the object LogisticRegression and give random_state =
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, Y_train)
```

Out[16]:
```
LogisticRegression(random_state=0)
```

In [21]:
```
Y_pred = classifier.predict(X_test)
```

In [22]:
```
Y_pred[0:9]
```

Out[22]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0], dtype=int64)
```

In [23]:
```
#Using Confusion matrix we can get accuracy of our model.

from sklearn.metrics import confusion_matrix
cnf_matrix = confusion_matrix(Y_test, Y_pred)
cnf_matrix
```

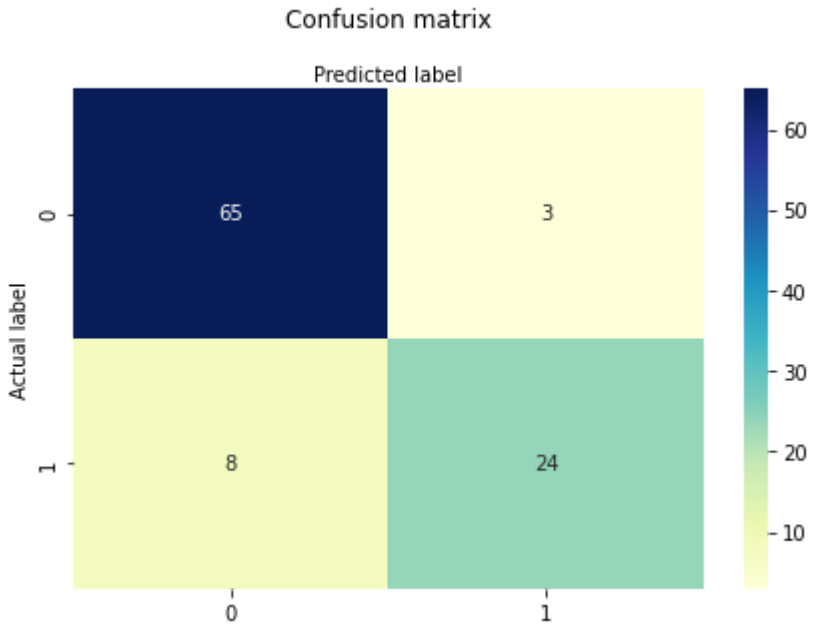
Out[23]:
```
array([[65,  3],
       [ 8, 24]], dtype=int64)
```

In [24]:
```
#Let's evaluate the model using model evaluation metrics such as accuracy, precision, a
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(Y_test, Y_pred))
print("Precision:",metrics.precision_score(Y_test, Y_pred))
print("Recall:",metrics.recall_score(Y_test, Y_pred))
```

```
Accuracy: 0.89
Precision: 0.8888888888888888
Recall: 0.75
```

In [25]:
```
#Let's visualize the results of the model in the form of a co#nfusion matrix using matp
#Here, you will visualize the confusion matrix using Heatmap.
import seaborn as sns
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

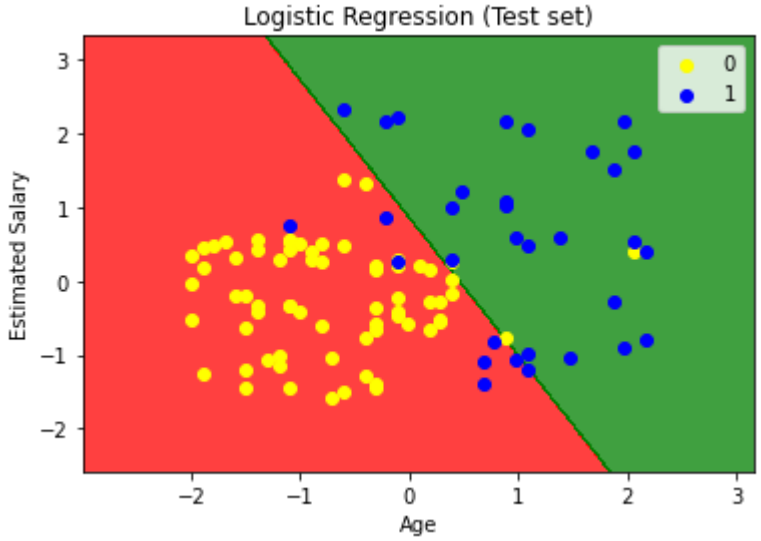
Out[25]:
```
Text(0.5, 257.44, 'Predicted label')
```



In [30]:
```
import warnings
```

```
warnings.filterwarnings('ignore')
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(X
             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(('yellow', 'blue'))(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 avoided a
s value-mapping will have precedence in case its length matches with *x* & *y*.  Please
use the *color* keyword-argument or provide a 2-D 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 avoided a
s value-mapping will have precedence in case its length matches with *x* & *y*.  Please
use the *color* keyword-argument or provide a 2-D array with a single row if you intend
to specify the same RGB or RGBA value for all points.
```



In [32]:
```
#Now let's try Naive Gaussian Bays
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, Y_train)
```

Out[32]:
```
GaussianNB()
```

In [42]:
```
Y2_pred  =  classifier.predict(X_test)
```

In [43]:
```
Y2_pred
```

Out[43]:
```
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, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1], dtype=int64)
```

In [44]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(Y_test, Y2_pred)
ac = accuracy_score(Y_test, Y2_pred)
```

In [45]:
```
cm
```

Out[45]:
```
array([[65,  3],
       [ 7, 25]], dtype=int64)
```

In [46]:
```
ac
```

Out[46]:
```
0.9
```

In [ ]:

In [ ]: