In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn import metrics
import seaborn as sns
import warnings
from matplotlib.colors import ListedColormap
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
```

In [2]:
```
Diabetus = pd.read_csv('C:/Users/andre/Desktop/ECGR HW/ECGR 4105/Diabetus.csv')
Diabetus.head()
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	O
0	6	148	72	35	0	33.6	0.627	50	
1	1	85	66	29	0	26.6	0.351	31	
2	8	183	64	0	0	23.3	0.672	32	
3	1	89	66	23	94	28.1	0.167	21	
4	0	137	40	35	168	43.1	2.288	33	

In [3]:
```
Diabetus.head(20)
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33
5	5	116	74	0	0	25.6	0.201	30
6	3	78	50	32	88	31.0	0.248	26
7	10	115	0	0	0	35.3	0.134	29
8	2	197	70	45	543	30.5	0.158	53
9	8	125	96	0	0	0.0	0.232	54
10	4	110	92	0	0	37.6	0.191	30

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
11	10	168	74	0	0	38.0	0.537	34
12	10	139	80	0	0	27.1	1.441	57
13	1	189	60	23	846	30.1	0.398	59
14	5	166	72	19	175	25.8	0.587	51
15	7	100	0	0	0	30.0	0.484	32
16	0	118	84	47	230	45.8	0.551	31
17	7	107	74	0	0	29.6	0.254	31
18	1	103	30	38	83	43.3	0.183	33
19	1	115	70	30	96	34.6	0.529	32

In [4]:
```
X = Diabetus.iloc[:, [0,1,2,3,4,5,6,7]].values
Y = Diabetus.iloc[:, 8].values
```

In [5]:
```
X[0:10]
```

Out[5]:
```
array([[6.000e+00, 1.480e+02, 7.200e+01, 3.500e+01, 0.000e+00, 3.360e+01,
        6.270e-01, 5.000e+01],
       [1.000e+00, 8.500e+01, 6.600e+01, 2.900e+01, 0.000e+00, 2.660e+01,
        3.510e-01, 3.100e+01],
       [8.000e+00, 1.830e+02, 6.400e+01, 0.000e+00, 0.000e+00, 2.330e+01,
        6.720e-01, 3.200e+01],
       [1.000e+00, 8.900e+01, 6.600e+01, 2.300e+01, 9.400e+01, 2.810e+01,
        1.670e-01, 2.100e+01],
       [0.000e+00, 1.370e+02, 4.000e+01, 3.500e+01, 1.680e+02, 4.310e+01,
        2.288e+00, 3.300e+01],
       [5.000e+00, 1.160e+02, 7.400e+01, 0.000e+00, 0.000e+00, 2.560e+01,
        2.010e-01, 3.000e+01],
       [3.000e+00, 7.800e+01, 5.000e+01, 3.200e+01, 8.800e+01, 3.100e+01,
        2.480e-01, 2.600e+01],
       [1.000e+01, 1.150e+02, 0.000e+00, 0.000e+00, 0.000e+00, 3.530e+01,
        1.340e-01, 2.900e+01],
       [2.000e+00, 1.970e+02, 7.000e+01, 4.500e+01, 5.430e+02, 3.050e+01,
        1.580e-01, 5.300e+01],
       [8.000e+00, 1.250e+02, 9.600e+01, 0.000e+00, 0.000e+00, 0.000e+00,
        2.320e-01, 5.400e+01]])
```

In [6]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size=0.8, test_size = 0
```

In [7]:
```
SS_X = StandardScaler()
X_train = SS_X.fit_transform(X_train)
X_test = SS_X.fit_transform(X_test)
```

In [8]:
```
Logistic_Class = LogisticRegression(random_state=0)
Logistic_Class.fit(X_train, Y_train)
```

Out[8]: LogisticRegression(random_state=0)

In [9]:
```
Y_pred = Logistic_Class.predict(X_test)
```

In [10]:
```
Y_pred[0:9]
```

Out[10]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0], dtype=int64)
```

In [11]:
```
cnf_matrix = confusion_matrix(Y_test, Y_pred)
cnf_matrix
```

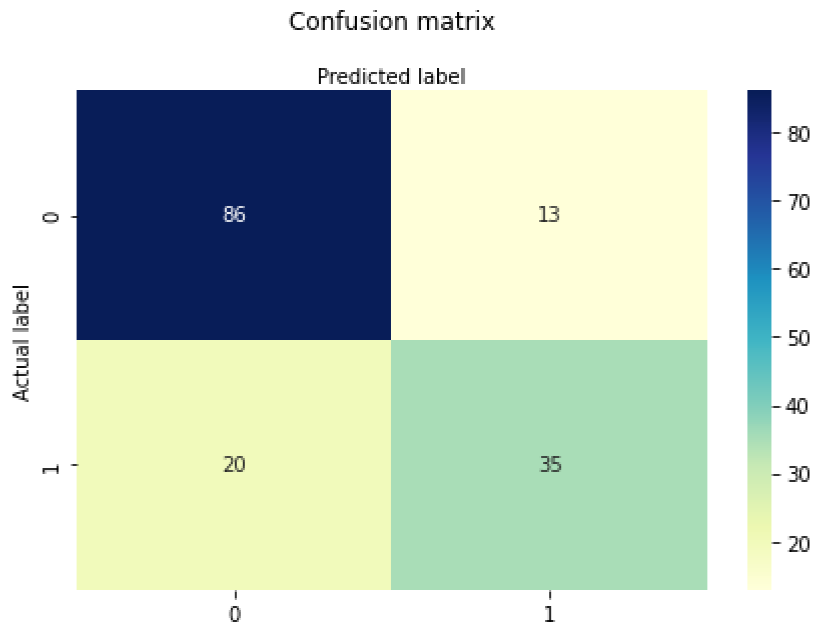
Out[11]:
```
array([[86, 13],
       [20, 35]], dtype=int64)
```

In [12]:
```
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: 0.7857142857142857
Precision: 0.7291666666666666
Recall: 0.6363636363636364
```

In [13]:
```
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[13]:
```
Text(0.5, 257.44, 'Predicted label')
```



In [14]:
```
#2-Color Plot (Not Working)

#warnings.filterwarnings('ignore')
#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,
 #                    np.arange(start = X_set[:, 1].min() -1, stop= X_set[:, 1].max()+1,

#plt.contourf(X1,X2, Logistic_Class.predict(np.array([X1.ravel(), X2.ravel()]).T).resha
   #          cmap = ListedColormap(('red','blue')))
  #      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','blue'))(i)
#plt.title('Logistic Regression(Test Set)')
#plt.xlabel('Age')
#plt.ylabel('Estimated Salary')
#plt.legend()
#plt.show()
```

In [ ]:

In [15]:
```
#Part 2: Gausian Bayes Thm
```

In [16]:
```
classifier = GaussianNB()
classifier.fit(X_train, Y_train)
```

Out[16]: GaussianNB()

In [17]:
```
Y2_pred  =  classifier.predict(X_test)
```

In [18]:
```
cm = confusion_matrix(Y_test, Y2_pred)
ac = accuracy_score(Y_test, Y2_pred)
```

In [19]:
```
print("Accuracy:",metrics.accuracy_score(Y_test, Y2_pred))
print("Precision:",metrics.precision_score(Y_test, Y2_pred))
print("Recall:",metrics.recall_score(Y_test, Y2_pred))
```

```
Accuracy: 0.7922077922077922
Precision: 0.7169811320754716
Recall: 0.6909090909090909
```

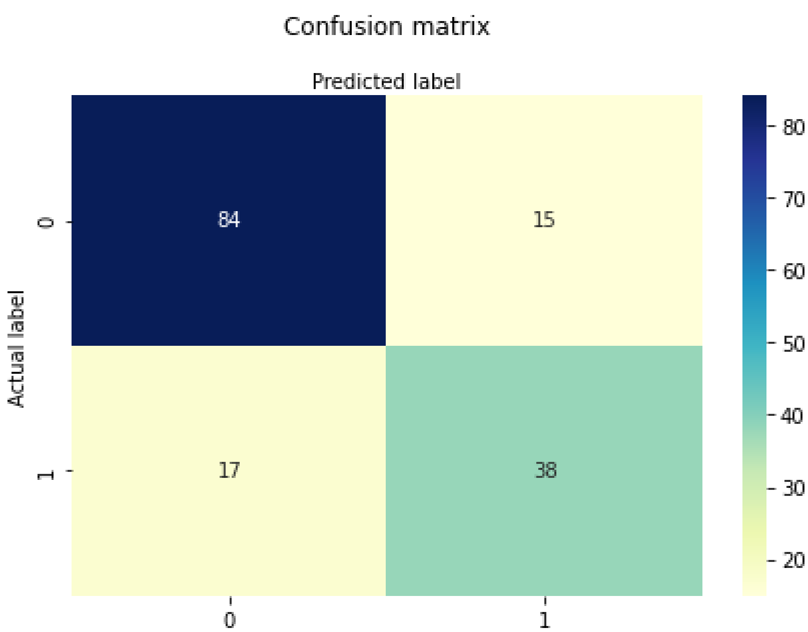
In [20]:
```
#Builds the matrix for problem 2
cnf_matrix = confusion_matrix(Y_test, Y2_pred)
cnf_matrix
```

Out[20]:
```
array([[84, 15],
       [17, 38]], dtype=int64)
```

In [21]:
```
#Confusion Matrix problem 2

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[21]:
```
Text(0.5, 257.44, 'Predicted label')
```



In [22]:

In [ ]:

In [ ]:

In [ ]:
```
#Part C: K-Fold of the Logistic Regression
```

In [23]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
```

In [24]:
```
#Code for K = 10 Folds
dataset = pd.read_csv('C:/Users/andre/Desktop/ECGR HW/ECGR 4105/Diabetus.csv')
X = dataset.iloc[:, [0, 7]]
Y = dataset.iloc[:, 8]
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,random_state =42)
#the train_test split function by default has a training (80%) and testing (20%) split
knnclassifier = KNeighborsClassifier(n_neighbos=10) #Number of folds
knnclassifier.fit(X_train,Y_train)
Y_pred = knnclassifier.predict(X_test)
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))
```

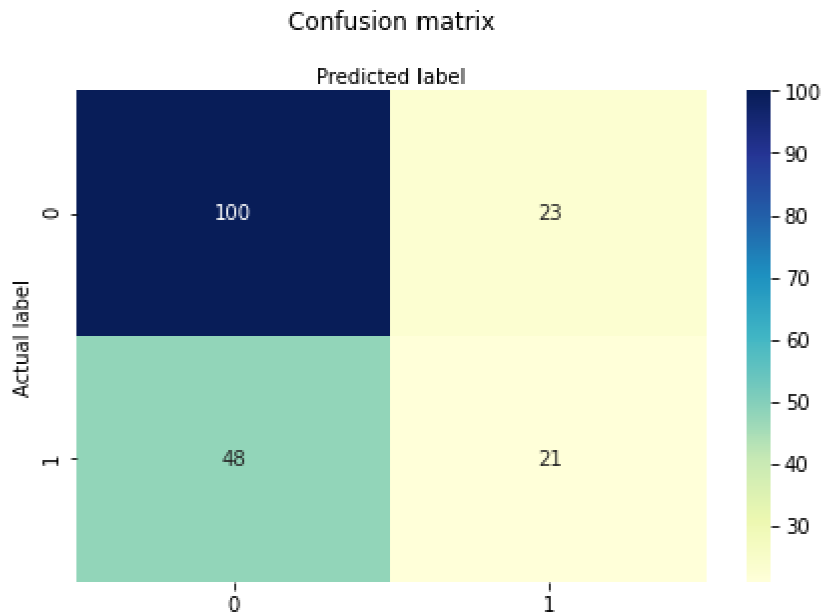
```
Accuracy: 0.6302083333333334
Precision: 0.4772727272727273
Recall: 0.30434782608695654
```

In [25]:
```
#Build the matrix for K = 10
cnf_matrix = confusion_matrix(Y_test, Y_pred)
cnf_matrix
```

Out[25]:
```
array([[100,  23],
       [ 48,  21]], dtype=int64)
```

In [26]:
```
#Confusion Matrix for Problem 3 (K = 10)
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[26]:
```
Text(0.5, 257.44, 'Predicted label')
```



In [ ]:

In [ ]:

In [ ]:

In [27]:

```
#Code for K = 5 folds
dataset = pd.read_csv('C:/Users/andre/Desktop/ECGR HW/ECGR 4105/Diabetus.csv')
X = dataset.iloc[:, [0, 7]]
Y = dataset.iloc[:, 8]
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,random_state =42)
knnclassifier = KNeighborsClassifier(n_neighbors=5) #Number of folds
knnclassifier.fit(X_train,Y_train)
Y_pred = knnclassifier.predict(X_test)
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: 0.609375
Precision: 0.4375
Recall: 0.30434782608695654
```

In [28]:

```
#Build the Matrix for K = 5
cnf_matrix = confusion_matrix(Y_test, Y_pred)
cnf_matrix
```

Out[28]:

```
array([[96, 27],
       [48, 21]], dtype=int64)
```

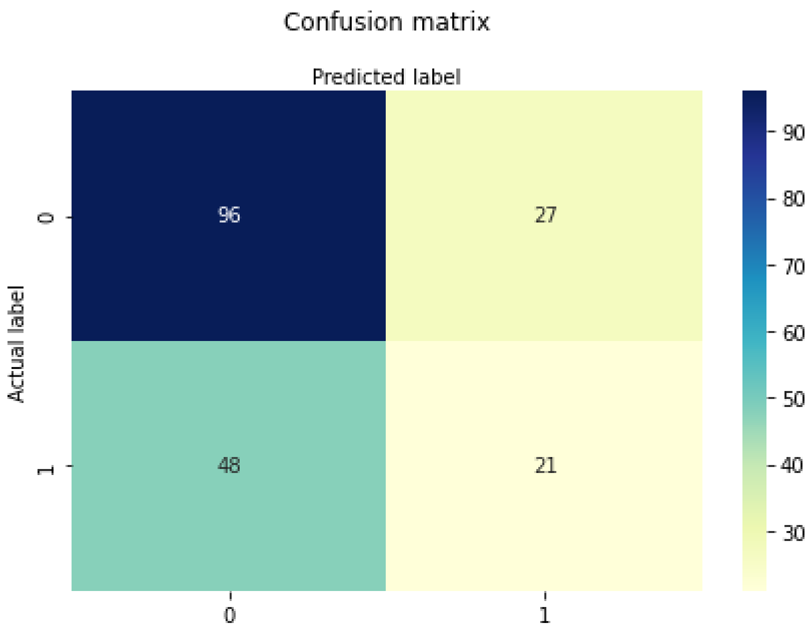
In [29]:

```
#Confusion Matrix for K = 5

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[29]: Text(0.5, 257.44, 'Predicted label')



In [30]:
```
#For K-Fold = 10
#When applying K-Fold Classification,  the Accuracy increased by a moderate ammount
#as well as the precision model. The significant differance is that the recall score
#doubled
```

In [31]:
```
#For K-fold = 5
#With K = 5, as expected form the previos trial, the accuracy and precision
#increased marginally from the initioal condition with no folds
#although these scores arent as large as the K=10 trial.
#But with the Recall score, there is not a significant difference between the trial wit
```

In [ ]:

In [ ]:

In [ ]:

In [32]:
```
#Part 4: K-Fold of the Nieve Bayes
#This does not make sense as applying K-Fold
#Nieve bayes is a clasifier for probablility
```

In [33]:
```
dataset = pd.read_csv('C:/Users/andre/Desktop/ECGR HW/ECGR 4105/Diabetus.csv')
X = dataset.iloc[:, [0, 7]]
Y = dataset.iloc[:, 8]
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,random_state =42)
#the train_test split function by default has a training (80%) and testing (20%) split
knnclassifier = KNeighborsClassifier(n_neighbors=10) #Number of folds
knnclassifier.fit(X_train,Y_train)
Y_pred2 = knnclassifier.predict(X_test)
print("Accuracy:",metrics.accuracy_score(Y_test, Y_pred2))
print("Precision:",metrics.precision_score(Y_test, Y_pred2))
print("Recall:",metrics.recall_score(Y_test, Y_pred2))
```

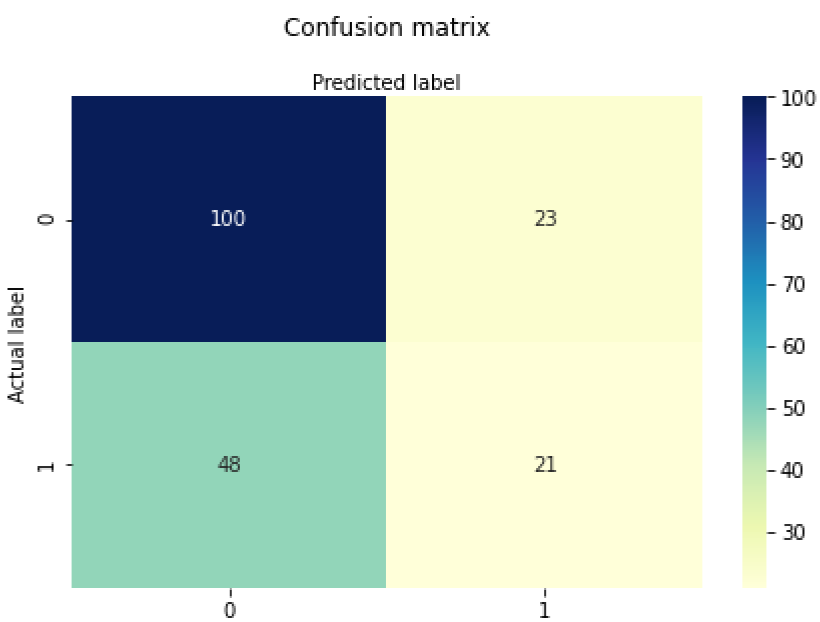
```
Accuracy: 0.6302083333333334
Precision: 0.4772727272727273
Recall: 0.30434782608695654
```

In [34]:
```
cnf_matrix = confusion_matrix(Y_test, Y_pred2)
cnf_matrix
```

Out[34]:
```
array([[100,  23],
       [ 48,  21]], dtype=int64)
```

In [35]:
```
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[35]:
```
Text(0.5, 257.44, 'Predicted label')
```



In [ ]: