In [33]:
```python
import pandas as pd
import numpy as np
```

In [34]:
```python
data = pd.read_csv("diabetes.csv")
data
```

Out[34]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

768 rows × 9 columns

In [35]:
```python
df = pd.DataFrame(data)
df.head()
```

Out[35]:

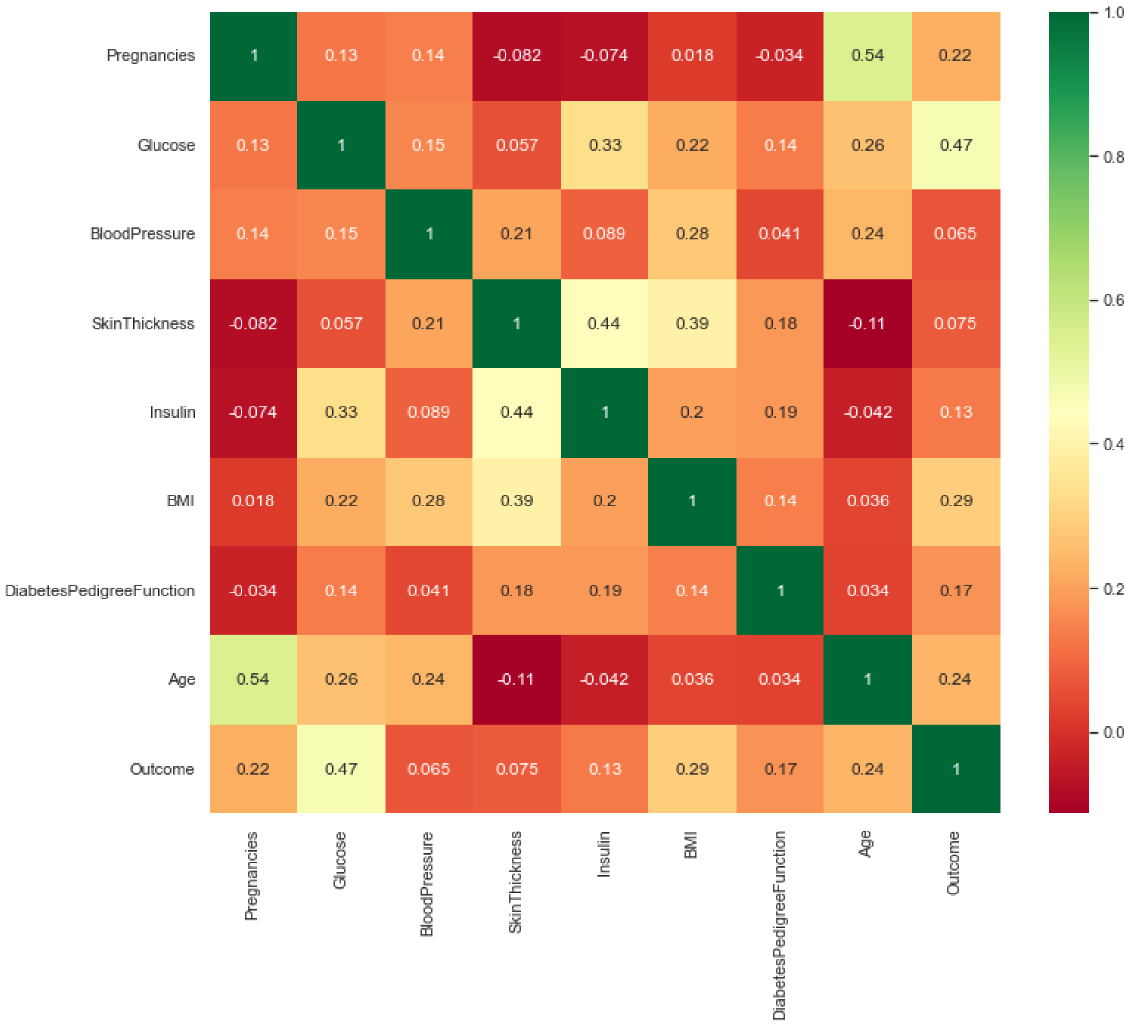
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct
0	6	148	72	35	0	33.6	0.
1	1	85	66	29	0	26.6	0.
2	8	183	64	0	0	23.3	0.
3	1	89	66	23	94	28.1	0.
4	0	137	40	35	168	43.1	2.

In [36]:
```python
df.isnull().sum()
```

Out[36]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [37]:

```python
plt.figure(figsize=(12,10))  # on this line I just set the size of figu
p=sns.heatmap(df.corr(), annot=True,cmap ='RdYlGn')  # seaborn has very
```



Manipulating and Cleaning our dataset

In [38]:

```python
cols_clean = ['Glucose','BloodPressure','SkinThickness','Insulin','BMI'
for i in cols_clean:
    df[i] = df[i].replace(0,np.NaN)
    cols_mean = int(df[i].mean(skipna=True))
    df[i] = df[i].replace(np.NaN, cols_mean)
data1 = df
data1.head().style.highlight_max(color="lightblue").highlight_min(color
```

Out[38]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesP
0	6	148.000000	72.000000	35.000000	155.000000	33.600000	
1	1	85.000000	66.000000	29.000000	155.000000	26.600000	
2	8	183.000000	64.000000	29.000000	155.000000	23.300000	
3	1	89.000000	66.000000	23.000000	94.000000	28.100000	
4	0	137.000000	40.000000	35.000000	168.000000	43.100000	

In [39]:

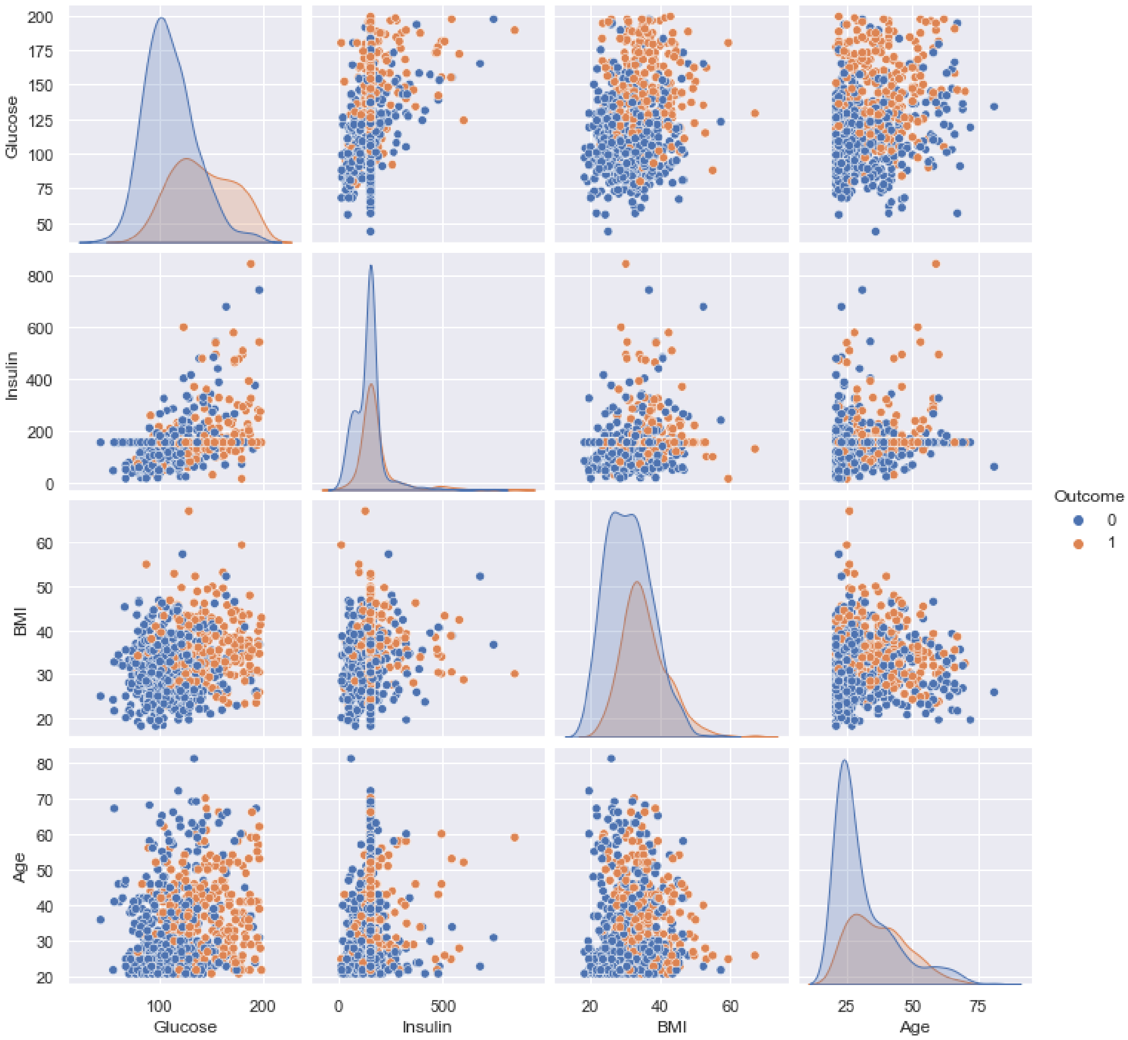
```
print(data1.describe())
```

```
       Pregnancies     Glucose  BloodPressure  SkinThickness    Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.00000
mean      3.845052  121.682292      72.386719      29.108073  155.28125
std       3.369578   30.435999      12.096642       8.791221   85.02155
min       0.000000   44.000000      24.000000       7.000000   14.00000
25%       1.000000   99.750000      64.000000      25.000000  121.50000
50%       3.000000  117.000000      72.000000      29.000000  155.00000
75%       6.000000  140.250000      80.000000      32.000000  155.00000
max      17.000000  199.000000     122.000000      99.000000  846.00000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    32.450911                  0.471876   33.240885    0.348958
std      6.875366                  0.331329   11.760232    0.476951
min     18.200000                  0.078000   21.000000    0.000000
25%     27.500000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

In [40]:

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# graph = ['Glucose','Insulin','BMI','Age','Outcome']
sns.set()
# print(sns.pairplot(data1[graph],hue='Outcome', diag_kind='kde'))
print(sns.pairplot(data1[graph],hue='Outcome', diag_kind='kde'))
```

```
<seaborn.axisgrid.PairGrid object at 0x000002A3E0B306A0>
```



In [46]:

```
# for the purpose of simplicity and analysing the most relevent  data ,
# Glucose , Insulin and BMI
# defining variables and features for the dataset for splitting
# q_cols = ['Glucose','Insulin','BMI','Outcome']
q_cols = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', '

df = data1[q_cols]
print(df.head(2))
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6    148.0           72.0           35.0    155.0  33.6
1            1     85.0           66.0           29.0    155.0  26.6

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
```

In [47]:

```
# # let's split the data into training and testing datasets
# split = 0.75 # 75% train and 25% test dataset
# total_len = len(df)
# split_df = int(total_len*split)
# train, test = df.iloc[:split_df,0:4],df.iloc[split_df:,0:4]
# train_x = train[['Glucose','Insulin','BMI']]
# train_y = train['Outcome']
# test_x = test[['Glucose','Insulin','BMI']]
# test_y = test['Outcome']

# Split the data into training and testing datasets
split = 0.75  # 75% train and 25% test dataset
total_len = len(df)
split_df = int(total_len * split)
train, test = df.iloc[:split_df], df.iloc[split_df:]

# Select the columns specified in q_cols for training and testing
train_x = train[q_cols[:-1]]  # Exclude the 'Outcome' column from featu
train_y = train['Outcome']    # Target variable
test_x = test[q_cols[:-1]]    # Exclude the 'Outcome' column from featu
test_y = test['Outcome']      # Target variable

```

In [48]:

```
a = len(train_x)
b = len(test_x)
print(' Training data =',a,'\n','Testing data =',b,'\n','Total data ler
```

```
 Training data = 576
 Testing data = 192
 Total data length =  768
```

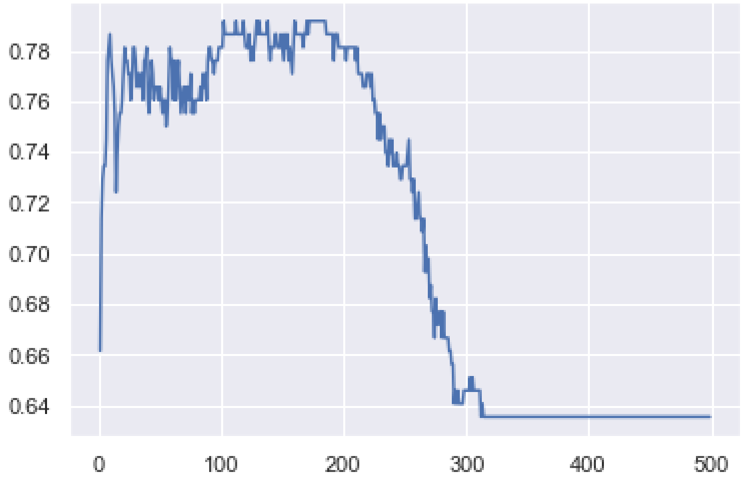
In [49]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

def knn(x_train, y_train, x_test, y_test,n):
    n_range = range(1, n)
    results = []
    for n in n_range:
        knn = KNeighborsClassifier(n_neighbors=n)
        knn.fit(x_train, y_train)
        #Predict the response for test dataset
        predict_y = knn.predict(x_test)
        accuracy = metrics.accuracy_score(y_test, predict_y)
        #matrix = confusion_matrix(y_test,predict_y)
        #seaborn_matrix = sns.heatmap(matrix, annot = True, cmap="Blues
        results.append(accuracy)
    return results
```

In [50]:

```
n= 500
output = knn(train_x,train_y,test_x,test_y,n)
n_range = range(1, n)
plt.plot(n_range, output)
```

Out[50]: [<matplotlib.lines.Line2D at 0x2a3ec0c6100>]



In [51]:

```
# best k that could optimize this model is between 100 to 200 offering
# ideal k value for this dataset should be 150 give or take
```

In [52]:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_scc
y_pred = knn(train_x,train_y,test_x,test_y,n)
cnf_matrix = confusion_matrix(test_y, y_pred)
```

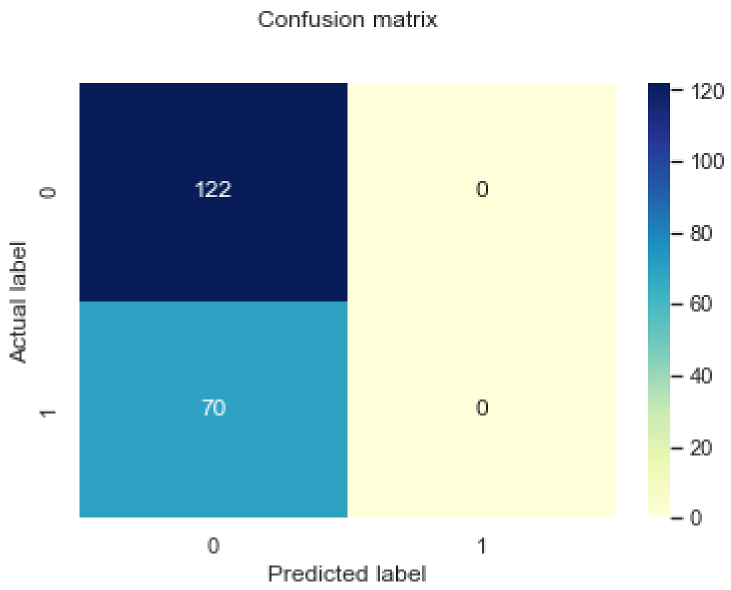
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call las
t)
~\AppData\Local\Temp/ipykernel_18924/597529570.py in <module>
      2 from sklearn.metrics import accuracy_score, precision_score, recal
l_score, f1_score, fbeta_score
      3 y_pred = knn(train_x,train_y,test_x,test_y,n)
----> 4 cnf_matrix = confusion_matrix(test_y, y_pred)

c:\users\kedar\appdata\local\programs\python\python39\lib\site-packages\sk
learn\metrics\_classification.py in confusion_matrix(y_true, y_pred, label
s, sample_weight, normalize)
    305     (0, 2, 1, 1)
    306     """
--> 307     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    308     if y_type not in ("binary", "multiclass"):
    309         raise ValueError("%s is not supported" % y_type)

c:\users\kedar\appdata\local\programs\python\python39\lib\site-packages\sk
learn\metrics\_classification.py in _check_targets(y_true, y_pred)
     82     y_pred : array or indicator matrix
     83     """
---> 84     check_consistent_length(y_true, y_pred)
     85     type_true = type_of_target(y_true, input_name="y_true")
     86     type_pred = type_of_target(y_pred, input_name="y_pred")

c:\users\kedar\appdata\local\programs\python\python39\lib\site-packages\sk
learn\utils\validation.py in check_consistent_length(*arrays)
    385     uniques = np.unique(lengths)
    386     if len(uniques) > 1:
--> 387         raise ValueError(
    388             "Found input variables with inconsistent numbers of sa
mples: %r"
    389             % [int(l) for l in lengths]

ValueError: Found input variables with inconsistent numbers of samples: [1
92, 499]
```

In [53]:

```
p = sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fm
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[53]:

```
Text(0.5, 12.5, 'Predicted label')
```



In [54]:

```
# Define your KNN function to return predictions
def knn2(x_train, y_train, x_test, y_test, n):
    knn = KNeighborsClassifier(n_neighbors=n)
    knn.fit(x_train, y_train)
    # Predict the response for the test dataset
    predict_y = knn.predict(x_test)
    return predict_y
```

In [55]:

```
n = 500
y_pred = knn2(train_x, train_y, test_x, test_y, n)
cnf_matrix = confusion_matrix(test_y, y_pred)
```

In [56]:

```
# Now you can calculate other metrics like accuracy, precision, recall,
accuracy = accuracy_score(test_y, y_pred)
precision = precision_score(test_y, y_pred)
recall = recall_score(test_y, y_pred)
f1 = f1_score(test_y, y_pred)
fbeta = fbeta_score(test_y, y_pred, beta=0.5)
```

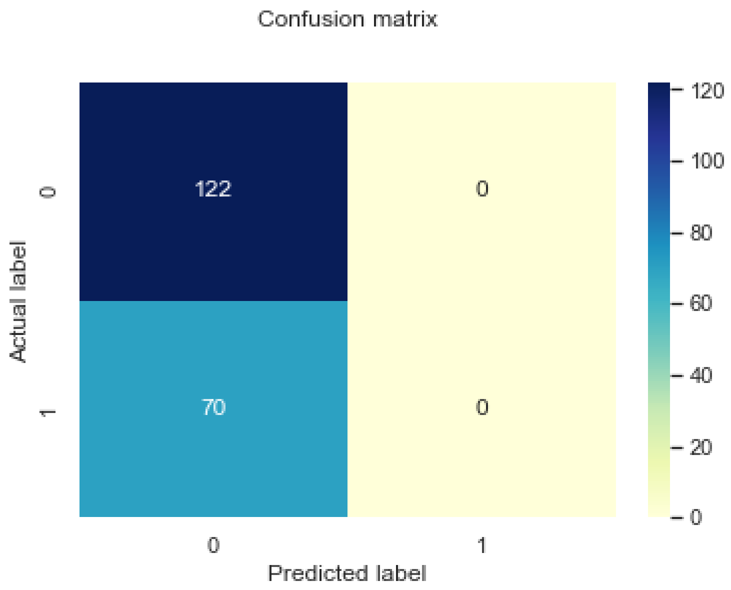
```
c:\users\kedar\appdata\local\programs\python\python39\lib\site-packages\sk
learn\metrics\_classification.py:1327: UndefinedMetricWarning: Precision i
s ill-defined and being set to 0.0 due to no predicted samples. Use `zero_
division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [57]:
```
# Print the confusion matrix and other metrics
print("Confusion Matrix:\n", cnf_matrix)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("F-beta Score:", fbeta)
```

```
Confusion Matrix:
 [[122   0]
 [ 70   0]]
Accuracy: 0.6354166666666666
Precision: 0.0
Recall: 0.0
F1 Score: 0.0
F-beta Score: 0.0
```

In [58]:
```
p = sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fm
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[58]:
```
Text(0.5, 12.5, 'Predicted label')
```



In [60]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classific

```

In [61]:

```python

# Load your dataset
# Replace 'your_dataset.csv' with the actual file path to your dataset
df = pd.read_csv('diabetes.csv')

# Define your feature columns and target column
q_cols = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', '
target_col = 'Outcome'

# Split the data into features (X) and target (y)
X = df[q_cols]
y = df[target_col]

# Split the data into training and testing datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2

# Perform feature scaling (standardization) on the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Create and train a K-nearest neighbors (KNN) classifier
k = 5  # You can adjust the value of k
knn_classifier = KNeighborsClassifier(n_neighbors=k)
knn_classifier.fit(X_train_scaled, y_train)

# Make predictions on the test data
y_pred = knn_classifier.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Print the results
print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(classification_rep)
```

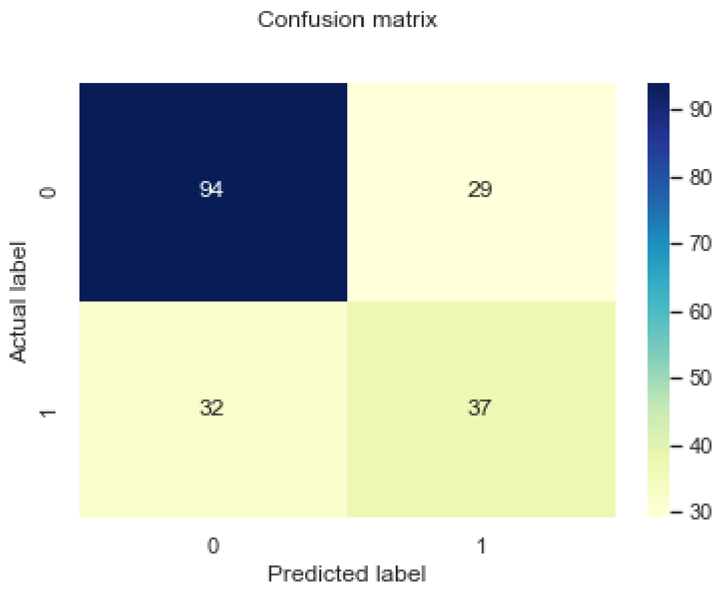
```
Accuracy: 0.6822916666666666
Confusion Matrix:
[[94 29]
 [32 37]]
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.76      0.76       123
           1       0.56      0.54      0.55        69

    accuracy                           0.68       192
   macro avg       0.65      0.65      0.65       192
weighted avg       0.68      0.68      0.68       192
```

In [63]:

```
p = sns.heatmap(pd.DataFrame(conf_matrix), annot=True, cmap="YlGnBu" ,f
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[63]:

```
Text(0.5, 12.5, 'Predicted label')
```



In [ ]:

```

```