Logistic Regression

Given the diabetes dataset from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. All patients here are females at least 21 years old Indian heritage. The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

Build a machine learning model to accurately predict whether or not the patients in the dataset have diabetes or not?

In [1]:
```
#import pandas
import pandas as pd

# load dataset
data = pd.read_csv("diabetes.csv")
```

In [2]:
```
data.head()
```

Out[2]:

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

In [3]:
```
#split dataset in features and target variable

X = data.iloc[:,0:8].values # Features
y = data.iloc[:,8].values # Target variable
```

In [4]:
```
# split X and y into training and testing sets
from sklearn import preprocessing
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_state=0)

# Use of feature scaling for quick convergence
# You may use minmax scaler or standard scaler

# min_max_scaler = preprocessing.MinMaxScaler()

# Using standard scaler for feature scaling

scaler = preprocessing.StandardScaler().fit(X)

X_train_standard = scaler.fit_transform(X_train)

X_test_standard = scaler.transform(X_test)
```

In [5]:
```
# import the class
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression()

# fit the model with data
logreg.fit(X_train_standard,y_train)

# Predict output for test set
y_pred=logreg.predict(X_test_standard)
```

In [6]:
```
# import the metrics class
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

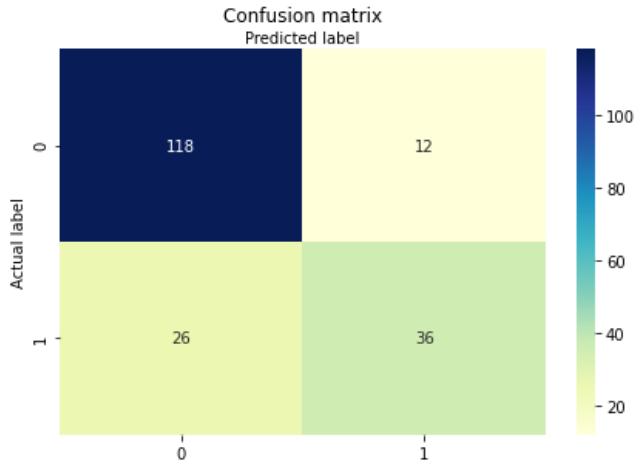
Out[6]:
```
array([[118,  12],
       [ 26,  36]], dtype=int64)
```

In [7]:
```
# visualize the confusion matrix using Heatmap.
```

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

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



In [8]:

```
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: 0.8020833333333334
Precision: 0.75
Recall: 0.5806451612903226
```

In [9]:

```
# Predict output for some unknown sample of the patient

#test_X = np.array([6,148,72,35,0,33.6,0.627,50])
#test_X = np.array([1,85,66,29,0,26.6,0.351,31])
test_X = np.array([1,101,42,36,0,28.1,0.512,21])

test_X_standard = scaler.transform(test_X.reshape(1, -1))
pred_y = logreg.predict(test_X_standard)
print("Prediction (whether patient has diabetes ?) = ", pred_y)
```

```
Prediction (whether patient has diabetes ?) =  [0]
```

In [ ]: