In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:
```
dataset = pd.read_csv('diabetes.csv')
```

In [3]:
```
dataset.head()
```

Out[3]:

	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 [4]:
```
dataset.head(20)
```

Out[4]:

	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
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1
10	4	110	92	0	0	37.6	0.191	30	0
11	10	168	74	0	0	38.0	0.537	34	1
12	10	139	80	0	0	27.1	1.441	57	0
13	1	189	60	23	846	30.1	0.398	59	1
14	5	166	72	19	175	25.8	0.587	51	1
15	7	100	0	0	0	30.0	0.484	32	1
16	0	118	84	47	230	45.8	0.551	31	1
17	7	107	74	0	0	29.6	0.254	31	1
18	1	103	30	38	83	43.3	0.183	33	0
19	1	115	70	30	96	34.6	0.529	32	1

In [5]:
```
#Problem 1
#We are considering all values in our dataset as variables
X = dataset.iloc[:, 0:8].values
Y = dataset.iloc[:,8].values
Y[0:10]
```

Out[5]:
```
array([1, 0, 1, 0, 1, 0, 1, 0, 1, 1], dtype=int64)
```

In [6]:
```
X[0:10]
```

Out[6]:
```
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 [7]:
```
#Splits dataset into Test and training data
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, random_state=0)
```

In [8]:
```
#feature scaling data
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
X_test[1]
```

Out[8]:
```
array([-0.54480808, -0.43719633,  0.24436264,  0.58457246,  0.15216202,
        0.17619533, -0.1876381 , -0.86242283])
```

In [9]:
```
# Split data into features and outcome

trainData = np.asarray(X_train[:, 0:8])
testData = np.asarray(X_test[:, 0:8])
trainData.shape
```

Out[9]:
```
(614, 8)
```

In [10]:
```
#imports Logistic regression function
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, Y_train)
```

Out[10]:
```
LogisticRegression(random_state=0)
```

In [11]:
```
LogisticRegression(random_state=0)
```

Out[11]:
```
LogisticRegression(random_state=0)
```

In [12]:
```
Y_pred = classifier.predict(X_test)
```

In [13]:
```
Y_pred[0:9]
```

Out[13]:
```
array([1, 0, 0, 1, 0, 0, 1, 1, 0], dtype=int64)
```

In [14]:
```
from sklearn.metrics import confusion_matrix
cnf_matrix = confusion_matrix(Y_test, Y_pred)
cnf_matrix
```

Out[14]:
```
array([[98,  9],
       [18, 29]], dtype=int64)
```

In [15]:
```
#Evaluates model using matrics accuracy, precision, recall and precision
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: 0.8246753246753247
Precision: 0.7631578947368421
Recall: 0.6170212765957447
```

In [16]:
```
#This function plots and visualizes the model in the form of a confusion matrix using matplot
import seaborn as cvs
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
cvs.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[16]:
```
Text(0.5, 257.44, 'Predicted label')
```


In [17]:
```
#Problem 2
#This funtion uses the Gaussian naive bayes model t
from sklearn.naive_bayes import GaussianNB
gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)
y_pred = gaussian.predict(X_test)
```

In [18]:
```
#Prints the confusion matrix to determine model
cnf_matrix = confusion_matrix(Y_test, y_pred)
print(cnf_matrix)
```

```
[[93 14]
 [18 29]]
```

In [19]:
```
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: 0.7922077922077922
Precision: 0.6744186046511628
Recall: 0.6170212765957447
```

In [20]:
```
class_names = [0, 1]
tick_marks = np.arange(len(class_names))
plt.subplots()
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
```

Out[20]:
```
([<matplotlib.axis.YTick at 0x23c2ca66e20>,
  <matplotlib.axis.YTick at 0x23c2ca66a00>],
 [Text(0, 0, '0'), Text(0, 1, '1')])
```


In [21]:
```
cvs.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap='YlGnBu', fmt='g')
plt.title('Confusion matrix')
plt.xlabel('Predicted class')
plt.ylabel('Actual class')
```

Out[21]:
```
Text(33.0, 0.5, 'Actual class')
```


In [30]:
```
# Problem 3
#K- fold cross validation with logistic regression k =5 fold
from sklearn.model_selection import train_test_split, KFold, cross_validate
from numpy import mean
from numpy import std
from sklearn.metrics import make_scorer
from sklearn.model_selection import cross_validate
metrics = ['accuracy', 'precision', 'recall']
cv5 = KFold(n_splits=5, random_state=1, shuffle=True)
scores = cross_validate(classifier, X, Y, scoring=metrics, cv=cv5, n_jobs=-1)


print("K-Fold 5 Splits")
print('Accuracy: %.4f , Recall: %.4f , Precision: %.4f' %(np.mean(scores['test_accuracy']), np.mean(scores['tes
```

```
K-Fold 5 Splits
Accuracy: 0.7630 , Recall: 0.5527 , Precision: 0.7047
```

In [34]:
```
cv10 = KFold(n_splits=10, random_state=1, shuffle=True)
scores = cross_validate(classifier, X, Y, scoring=metrics, cv=cv10, n_jobs=-1)


print("K-Fold 10 Splits")
print('Accuracy: %.4f , Recall: %.4f , Precision: %.4f' %(np.mean(scores['test_accuracy']), np.mean(scores['tes
```

```
K-Fold 10 Splits
Accuracy: 0.7721 , Recall: 0.5543 , Precision: 0.7260
```

In [35]:
```
#Problem 4
#In this section we use K-FOLD cross validation of Gaussian naive bayes model k = 5
scores = cross_validate(gaussian, X, Y, scoring=metrics, cv=cv5, n_jobs=-1)

print("K-Fold 5 Splits")
print('Accuracy: %.4f , Recall: %.4f , Precision: %.4f' %(np.mean(scores['test_accuracy']), np.mean(scores['tes
```

```
K-Fold 5 Splits
Accuracy: 0.7579 , Recall: 0.6013 , Precision: 0.6710
```

In [36]:
```
#This function performs K-fold cross-validation for training and validation.
scores = cross_validate(gaussian, X, Y, scoring=metrics, cv=cv10, n_jobs=-1)

print("K-Fold 10 Splits")
print('Accuracy: %.4f , Recall: %.4f , Precision: %.4f' %(np.mean(scores['test_accuracy']), np.mean(scores['tes
```

```
K-Fold 10 Splits
Accuracy: 0.7565 , Recall: 0.6009 , Precision: 0.6652
```

In [ ]:
```

```

In [ ]:
```

```