4105 Hw 2

In [27]:

```
import numpy as np
import pandas as pd
import nbconvert
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.decomposition import PCA
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
import warnings
warnings.filterwarnings('ignore')
from sklearn.datasets import load_breast_cancer


from sklearn import model_selection
from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn import metrics
```

In [28]:

```
cancer = load_breast_cancer()
Data=cancer.data
Data.shape
```

Out[28]:

```
(569, 30)
```

In [29]:

```
x=pd.DataFrame(Data)
x.head()
```

Out[29]:

	0	1	2	3	4	5	6	7	8	9	...	20	2
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.3
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.4
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.5
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.5
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.6

5 rows × 30 columns

In [30]:

```
Target= cancer.target
y=pd.DataFrame(Target)
y.head()
```

Out[30]:

	0
0	0
1	0
2	0
3	0
4	0

In [31]:

```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, train_size = 0.8, test_size
```

In [32]:

```
scaler = StandardScaler()
xtrain = scaler.fit_transform(xtrain)
xtest = scaler.transform(xtest)
xtrain
```

Out[32]:

```
array([[-1.15036482, -0.39064196, -1.12855021, ..., -0.75798367,
        -0.01614761, -0.38503402],
       [-0.93798972,  0.68051405, -0.94820146, ..., -0.60687023,
         0.09669004, -0.38615797],
       [ 0.574121  , -1.03333557,  0.51394098, ..., -0.02371948,
        -0.20050207, -0.75144254],
       ...,
       [-1.32422924, -0.20048168, -1.31754581, ..., -0.97974953,
        -0.71542314, -0.11978123],
       [-1.24380987, -0.2245526 , -1.28007609, ..., -1.75401433,
        -1.58157125, -1.00601779],
       [-0.73694129,  1.14989702, -0.71226578, ..., -0.27460457,
        -1.25895095,  0.21515662]])
```

In [36]:

```
dataset = datasets.load_iris()
model = GaussianNB()
model.fit(xtrain, ytrain)

expected = ytrain
predicted = model.predict(xtrain)

print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       0.95      0.90      0.92       165
           1       0.94      0.98      0.96       290

    accuracy                           0.95       455
   macro avg       0.95      0.94      0.94       455
weighted avg       0.95      0.95      0.95       455

[[148  17]
 [  7 283]]
```

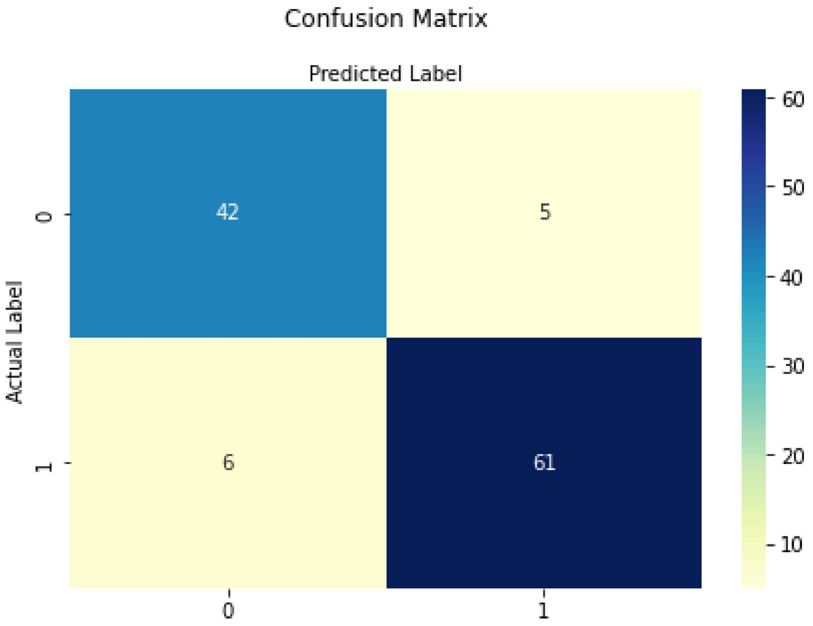
In [41]:

```
preds = model.predict(xtest)
print(classification_report(ytest,preds))
matrix = confusion_matrix(ytest,preds)
fig,ax = plt.subplots()
sns.heatmap(pd.DataFrame(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')
```

```
              precision    recall  f1-score   support

           0       0.88      0.89      0.88        47
           1       0.92      0.91      0.92        67

    accuracy                           0.90       114
   macro avg       0.90      0.90      0.90       114
weighted avg       0.90      0.90      0.90       114
```

Out[41]:

```
Text(0.5, 257.44, 'Predicted Label')
```



Problem 2

In [43]:
```
scaler = preprocessing.StandardScaler()
x_train = scaler.fit_transform(xtrain)
x_test = scaler.fit_transform(xtest)
x_train = pd.DataFrame(x_train)
x_train.head()
```

Out[43]:

	0	1	2	3	4	5	6	7	8
0	-1.150365	-0.390642	-1.128550	-0.958764	0.310984	-0.595995	-0.802596	-0.802490	0.294539
1	-0.937990	0.680514	-0.948201	-0.821525	-0.609636	-0.909867	-0.660669	-0.898716	0.754935
2	0.574121	-1.033336	0.513941	0.408586	-0.106161	-0.363019	-0.417990	-0.088446	-0.271820
3	-0.547220	-0.316022	-0.577622	-0.566615	0.586662	-0.649331	-0.805298	-0.500065	0.331078
4	-0.527398	0.791240	-0.561563	-0.523571	-1.051446	-1.017532	-0.905149	-0.935806	-0.969721

5 rows × 30 columns

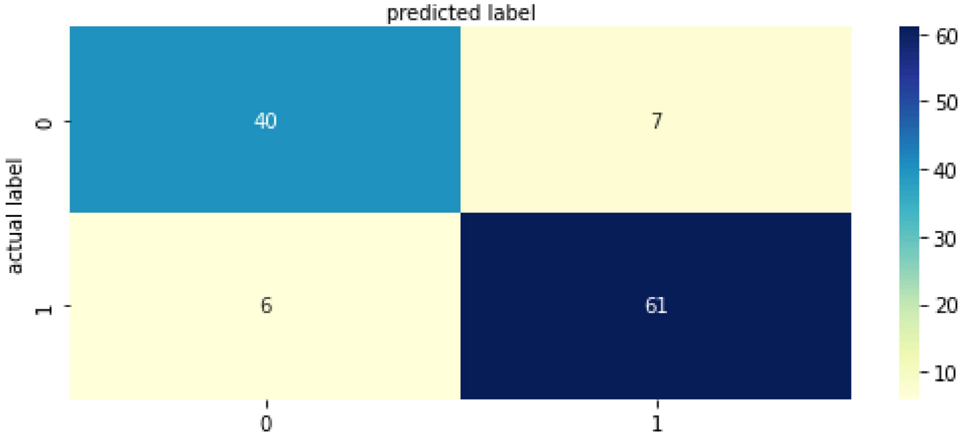
In [86]:

```
K = [1,2,3,4,5,10,13,15]
for i in range(len(K)):
    pca = PCA(n_components=K[i], svd_solver = "auto")
    PCAxTrain = pca.fit_transform(xtrain)
    pca = PCA(n_components=K[i], svd_solver = "auto")
    PCAxTest = pca.fit_transform(xtest)
    clf = LogisticRegression(C=10,solver='liblinear')
    clf.fit(x_train_pca,ytrain)
    preds = clf.predict(x_test_pca)
    print("K = "+str(K[i])+":")
    print(classification_report(ytest,preds))
    marix = confusion_matrix(ytest,preds)
    fig,ax = plt.subplots()
    sns.heatmap(pd.DataFrame(matrix),annot=True,cmap="YlGnBu",fmt='g')
    ax.xaxis.set_label_position('top')
    plt.title('confusion matrix (key: 0 = benign 1 = malignant) at k = ' + str(K
    plt.ylabel('actual label')
    plt.xlabel('predicted label')
```

```
           0       0.88      0.89      0.88        47
           1       0.92      0.91      0.92        67

    accuracy                           0.90       114
   macro avg       0.90      0.90      0.90       114
weighted avg       0.90      0.90      0.90       114

K = 10:
              precision    recall  f1-score   support

           0       0.88      0.89      0.88        47
           1       0.92      0.91      0.92        67

    accuracy                           0.90       114
   macro avg       0.90      0.90      0.90       114
weighted avg       0.90      0.90      0.90       114

K = 13:
              precision    recall  f1-score   support
```

In [87]:

```
K = [1,2,3,4,5,10,13,15]
for i in range(len(K)):
    pca = PCA(n_components=K[i], svd_solver = "auto")
    PCAxTrain = pca.fit_transform(xtrain)
    pca = PCA(n_components=K[i], svd_solver = "auto")
    PCAxTest = pca.fit_transform(xtest)
    naive_bayes = GaussianNB()
    naive_bayes.fit(PCAxTrain,ytrain)
    preds = naive_bayes.predict(PCAxTest)

    print("K = "+str(K[i])+":")
    print("Accuracy:", metrics.accuracy_score(ytest, preds))
    print("Precision:", metrics.precision_score(ytest, preds))
    print("Recall:", metrics.recall_score(ytest, preds))
    print("")

    matrix = confusion_matrix(ytest,preds)
    fig,ax = plt.subplots()
    sns.heatmap(pd.DataFrame(matrix),annot=True,cmap="YlGnBu",fmt='g')
    ax.xaxis.set_label_position('top')
    plt.title('confusion matrix (key: 0 = benign 1 = malignant) at k = ' + str(K
    plt.xlabel('predicted label')
    plt.ylabel('actual label')
```



confusion matrix (key: 0 = benign 1 = malignant) at k = 1

confusion matrix (key: 0 = benign 1 = malignant) at k = 2

In [ ]: