In [176…
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.datasets import load_breast_cancer
```

In [171…
```
breast = load_breast_cancer()
```

In [30]:
```
breast_data = breast.data
breast_data.shape
```

Out[30]: (569, 30)

In [31]:
```
breast_input = pd.DataFrame(breast_data)
breast_input.head()
```

Out[31]:

	0	1	2	3	4	5	6	7	8	9	...	20	21	
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152

5 rows × 30 columns

In [32]:
```
breast_labels = breast.target
```

In [33]:
```
breast_labels.shape
```

Out[33]: (569,)

In [34]:
```
labels = np.reshape(breast_labels,(569,1))
```

In [35]:
```
final_breast_data = np.concatenate([breast_data,labels],axis=1)
```

In [36]:
```
final_breast_data.shape
```

Out[36]: (569, 31)

In [37]:
```
breast_dataset = pd.DataFrame(final_breast_data)
```

In [38]:
```
features = breast.feature_names
features
```

Out[38]:
```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

In [39]:
```
features_labels = np.append(features,'label')
```

In [40]:
```
breast_dataset.columns = features_labels
```

In [41]:
```
breast_dataset.head()
```

Out[41]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	m fra dimen
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05

5 rows × 31 columns

In [42]:
```
#1 Logistic Regression
a = breast_data.shape[1]
X = breast_dataset.values[:, :a]
y = breast_dataset.values[:,a]
```

In [43]:
```
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)
```

In [44]:
```
from sklearn.preprocessing import StandardScaler
S_X = StandardScaler()
X_train = S_X.fit_transform(X_train)
X_test = S_X.transform(X_test)
```

In [45]:
```
from sklearn.linear_model import LogisticRegression
```

```
classifier = LogisticRegression()
classifier.fit(X_train, y_train)
```

Out[45]:
```
LogisticRegression()
```

In [46]:
```
model = LogisticRegression()
```

In [47]:
```
model.fit(X_train, y_train)
```

Out[47]:
```
LogisticRegression()
```

In [48]:
```
model.predict(X_test)
```

Out[48]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 1., 0., 1., 0., 1., 0., 0.,
       1., 0., 1., 1., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 0., 1.,
       0., 1., 1., 0., 0., 0., 1., 1., 0., 0., 1., 0., 0., 1., 1., 1., 1.,
       1., 1., 1., 0., 0., 0., 1., 1., 0., 1., 1., 0., 1., 1., 1., 0., 0.,
       1., 1., 1., 0., 1., 1., 1., 0., 1., 0., 0., 1., 1., 1., 0., 1., 1.,
       1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 0., 1., 1., 1., 0., 1.,
       1., 1., 0., 1., 0., 1., 1., 1., 1., 0., 1., 1.])
```

In [49]:
```
model.score(X_test,y_test)
```

Out[49]:
```
0.9736842105263158
```

In [50]:
```
y_pred = classifier.predict(X_test)
y_pred[0:4]
```

Out[50]:
```
array([1., 1., 1., 1.])
```

In [51]:
```
from sklearn.metrics import confusion_matrix
cf_matrix = confusion_matrix(y_test, y_pred)
cf_matrix
```

Out[51]:
```
array([[36,  1],
       [ 2, 75]], dtype=int64)
```

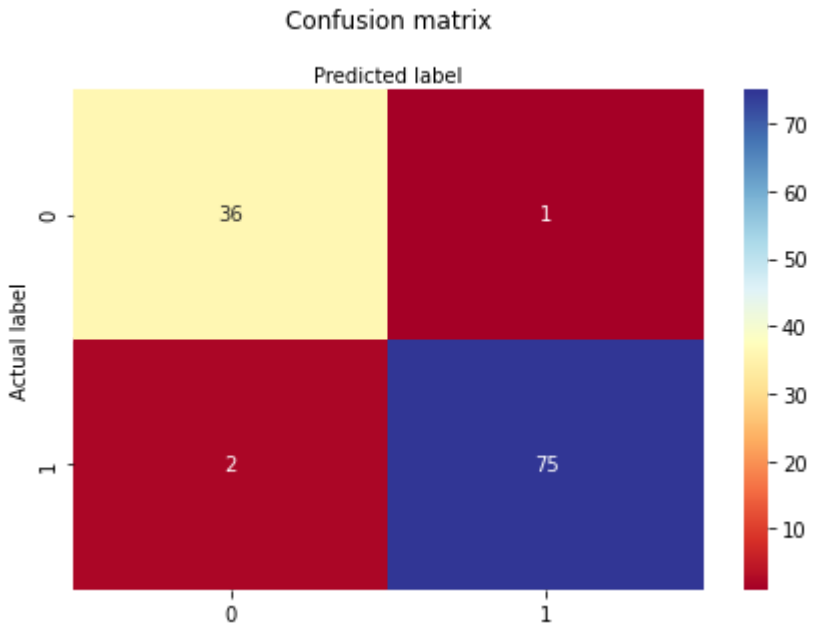
In [52]:
```
from sklearn import metrics
# Finding the Accuracy
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
print("Precision", metrics.precision_score(y_test, y_pred))
print("Recall" ,metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.9736842105263158
Precision 0.9868421052631579
Recall 0.974025974025974
```

In [124…
```
import seaborn as sns
class_names=[0,1]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
```

```
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
#Create map
sns.heatmap(pd.DataFrame(cf_matrix), annot=True,cmap="RdYlBu", 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[124… Text(0.5, 257.44, 'Predicted label')



In [88]:
```
from sklearn.preprocessing import StandardScaler
features = ['sepal length', 'sepal width', 'petal length', 'petal width']
```

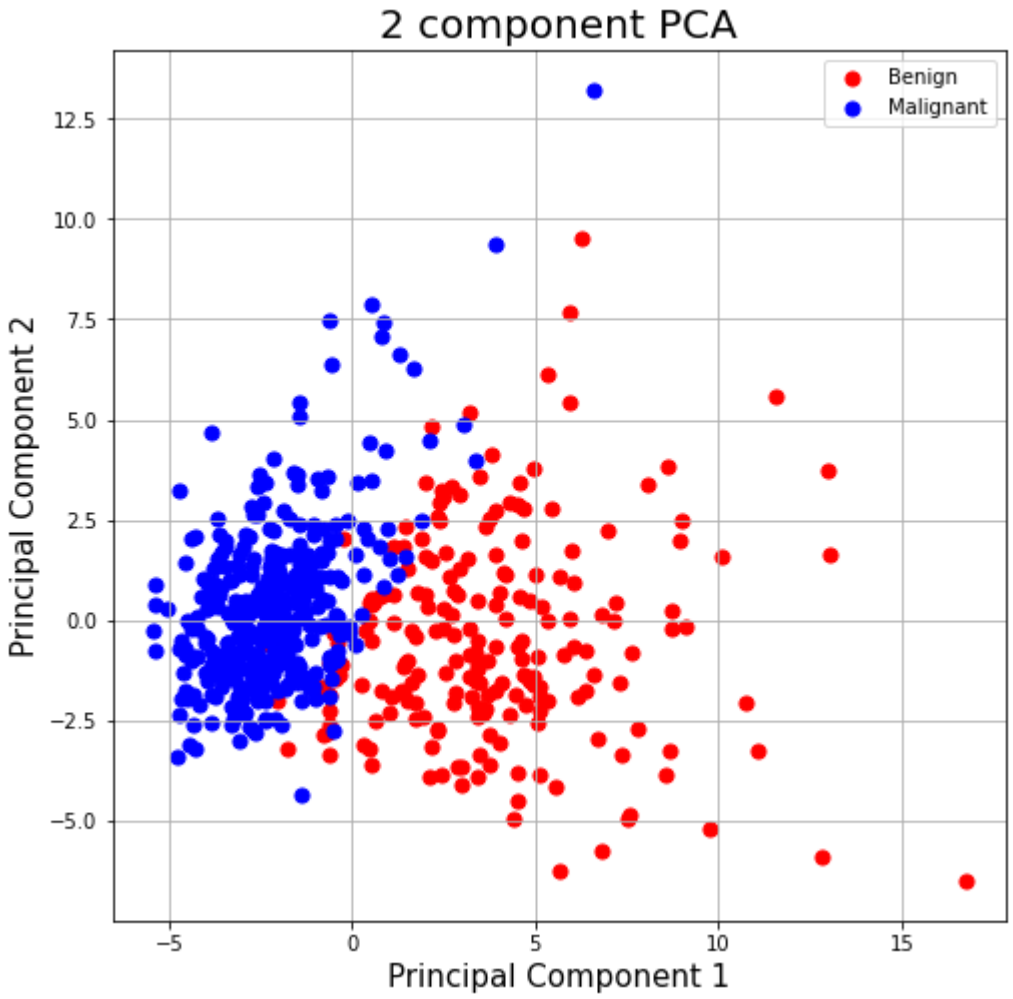
In [108…
```
# Problem 2 PCA
pca = PCA(n_components = 2)
pcs = pca.fit_transform(X)

X = breast_dataset.iloc[:,0:29].values
y = breast_dataset.iloc[:,30].values
X = StandardScaler().fit_transform(X)
```

In [109…
```
from sklearn.decomposition import PCA
principalDf = pd.DataFrame(data = pcs
             , columns = ['principal component 1', 'principal component 2'])
```

In [150…
```
finalDataFrame = pd.concat([principalDf, breast_dataset[['label']]], axis = 1)
```

In [151…
```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
```

```
ax.set_title('2 component PCA', fontsize = 20)
labels = ['Benign', 'Malignant']
colors = ['r', 'b']
for label, color in zip(targets,colors):
    indicesToKeep = finalDataFrame['label'] == label
    ax.scatter(finalDataFrame.loc[indicesToKeep, 'principal component 1']
               , finalDataFrame.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
    ax.legend(labels)
    ax.grid(labels)
```



In [179…

```
# Problem 3 LDA Feature Extraction Bayes Classifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import confusion_matrix
```

In [180…

```
from sklearn.datasets import load_wine
data = load_wine()
X = data.data
y = data.target
```

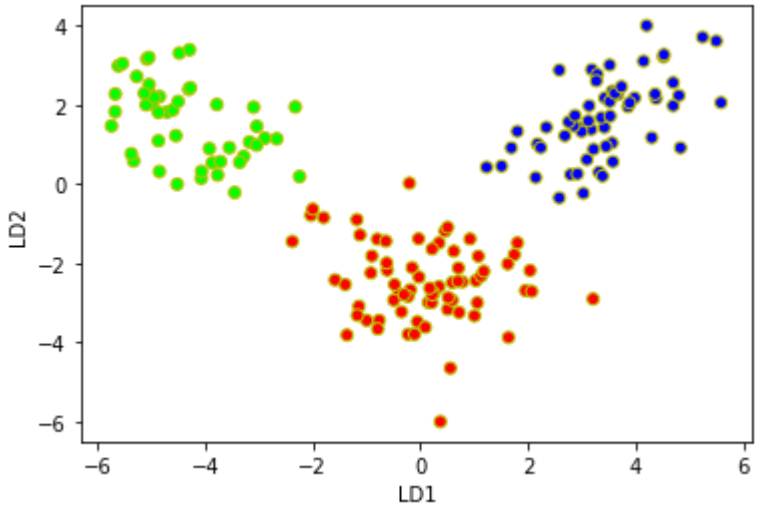
In [181…

```
lda = LinearDiscriminantAnalysis(n_components=2)
lda_1 = lda.fit_transform(X,y)
```

In [182…

```
plt.xlabel('LD1')
plt.ylabel('LD2')
plt.scatter(lda_1[:,0],lda_1[:,1], c=y,cmap='brg',edgecolors='y')
```

Out[182… <matplotlib.collections.PathCollection at 0x2c87c3a6af0>



In [185…

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
lda.fit(X_train,y_train)
y_pred = lda.predict(X_test)
print(accuracy_score(y_test,y_pred))
```

```
0.9722222222222222
```

In [ ]:

```
#4 Repeat problem 3, replace Bayes classifier with logistic regression.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: