In [381…

```
# Mubeen Quadrt
# Student Id: 801064313
# Homework 3
# Reference: Cancer Dataset PDF
# https://github.com/MubeenQ/Homework3/blob/main/MubeenQuadrtIntroToMLHomework3.ipynb

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.datasets import load_breast_cancer
```

In [382…

```
breast = load_breast_cancer()
```

In [383…

```
breast_data = breast.data
breast_data.shape
```

Out[383… (569, 30)

In [384…

```
breast_input = pd.DataFrame(breast_data)
breast_input.head()
```

Out[384…

	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 [385…

```
breast_labels = breast.target
```

In [386…

```
breast_labels.shape
```

Out[386… (569,)

In [387…

```
labels = np.reshape(breast_labels,(569,1))
```

In [388…

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

In [389…

```
final_breast_data.shape
```

Out[389… (569, 31)

In [390…

```
breast_dataset = pd.DataFrame(final_breast_data)
features = breast.feature_names
features
```

Out[390…

```
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 [391…

```
features_labels = np.append(features,'label')
```

In [392…

```
breast_dataset.columns = features_labels
```

In [393…

```
breast_dataset.head()
```

Out[393…

	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 [394…

```
breast_dataset['label'].replace(0, 'Benign',inplace=True)
breast_dataset['label'].replace(1, 'Malignant',inplace=True)
```

In [395…

```
breast_dataset.tail()
```

Out[395…

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	dim
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	dim
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	C
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	C
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	C

5 rows × 31 columns

In [396…

```
# For the evaluation of this homework across all problems, use 80%, 20% split.

RSTATE = 0

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(breast_input, labels, test_size = 0
```

In [397…

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

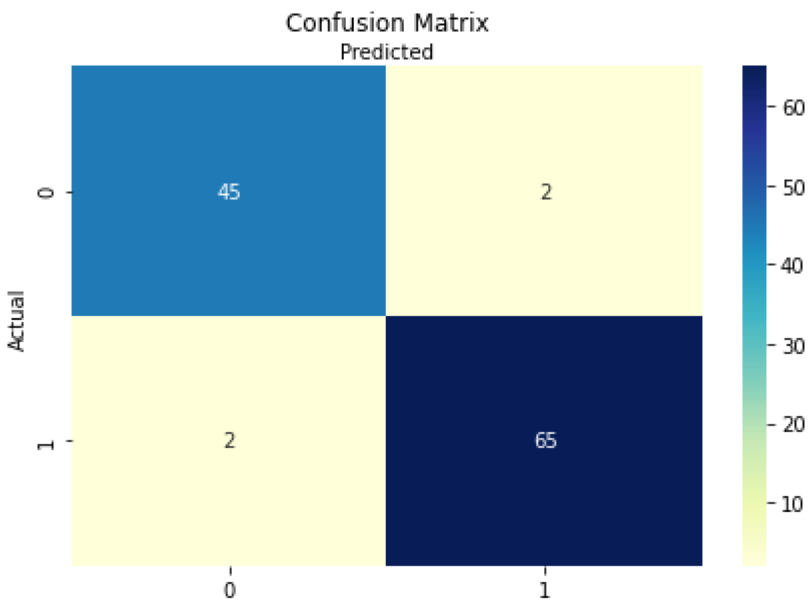
In [398…

```
# Question 1: Use the cancer dataset to build a logistic regression model
# to classify the type of cancer (Malignant vs. benign).
# First, create a logistic regression that takes all 30 input features for classificati

import warnings

from sklearn.datasets import load_breast_cancer

from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=RSTATE)
classifier.fit(X_train, Y_train)

Y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
cnf_matrix = confusion_matrix(Y_test, Y_pred)
print(cnf_matrix)
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))

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)

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')
plt.xlabel('Predicted')
plt.ylabel('Actual')
```

```
[[45  2]
 [ 2 65]]
Accuracy: 0.9649122807017544
Precision: 0.9701492537313433
Recall: 0.9701492537313433
```

Out[398… Text(33.0, 0.5, 'Actual')

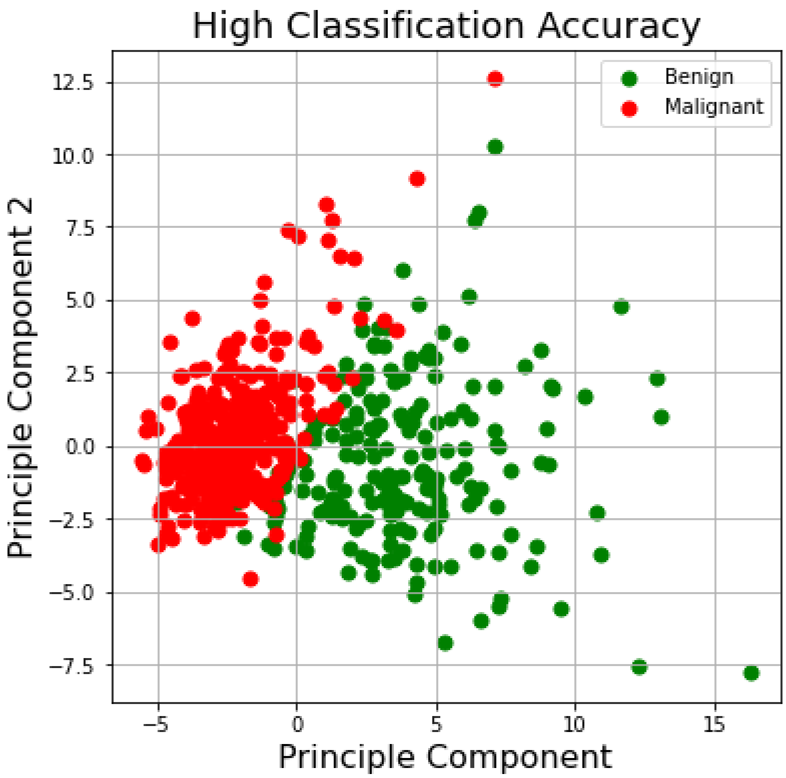


In [399…

```
# Question 2: Repeat problem 1, but this time use the PCA feature extraction for your t
# Perform N number of independent training (N=1, …, K).
# Identify the optimum number of K, principle components that achieve the highest class
# Plot your classification accuracy, precision, and recall over a different number of K

from sklearn.decomposition import PCA

X = StandardScaler().fit_transform(breast_input)
Y = breast_labels

K = np.empty([30,1])
Accuracy = np.empty([30,1])
Precision = np.empty([30,1])
Recall = np.empty([30,1])

print("Using PCA Feature Extraction:")

for n in range(1, 31):
    pca = PCA(n_components=n)
    PrinComp = pca.fit_transform(X)

    if n == 2:
        PrinDF = pd.DataFrame(data = PrinComp
                      , columns = ['Principle Component', 'Principle Component 2'])
        finalDf = pd.concat([PrinDF, breast_dataset[['label']]], axis = 1)

        fig = plt.figure(figsize = (6,6))
        ax = fig.add_subplot(1,1,1)
        ax.set_xlabel('Principle Component', fontsize = 16)
```

```
        ax.set_ylabel('Principle Component 2', fontsize = 16)
        ax.set_title('High Classification Accuracy', fontsize = 18)
        targets = ['Benign', 'Malignant']
        colors = ['g', 'r']
        for target, color in zip(targets,colors):
            indicesToKeep = finalDf['label'] == target
            ax.scatter(finalDf.loc[indicesToKeep, 'Principle Component']
                       , finalDf.loc[indicesToKeep, 'Principle Component 2']
                       , c = color
                       , s = 50)
        ax.legend(targets)
        ax.grid()

    X_train, X_test, Y_train, Y_test = train_test_split(PrinComp, Y, test_size = 0.2, r
    classifier = LogisticRegression(random_state=RSTATE)
    classifier.fit(X_train, Y_train)

    Y_pred = classifier.predict(X_test)
    from sklearn.metrics import confusion_matrix
    cnf_matrix = confusion_matrix(Y_test, Y_pred)

    from sklearn import metrics

    K[n-1] = n
    Accuracy[n-1] = metrics.accuracy_score(Y_test, Y_pred)
    Precision[n-1] = metrics.precision_score(Y_test, Y_pred)
    Recall[n-1] = metrics.recall_score(Y_test, Y_pred)

    MAccuracy = 0.0
    MPrecision = 0.0
    MRecall = 0.0
    MAccuracy_K = 0
    MPrecision_K = 0
    MRecall_K = 0

    if MAccuracy < np.amax(Accuracy):
        MAccuracy = np.amax(Accuracy);
        MAccuracy_K = n;

    if MPrecision < np.amax(Precision):
        MPrecision = np.amax(Precision);
        MPrecision_K = n;

    if MRecall < np.amax(Recall):
        MRecall = np.amax(Recall);
        MRecall_K = n;
```

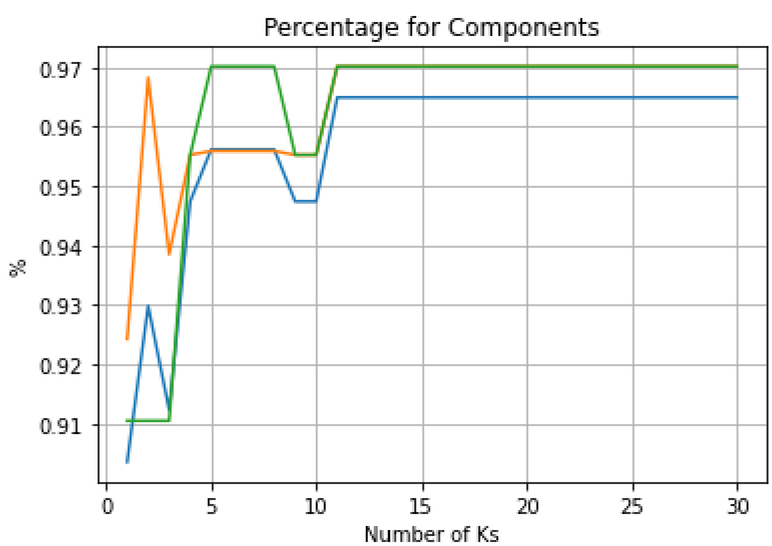
Using PCA Feature Extraction:



In [405…

```
plt.title('Percentage for Components')
plt.ylabel('%')
plt.xlabel('Number of Ks')

plt.grid()
plt.plot(K, Accuracy)
plt.plot(K, Precision)
plt.plot(K, Recall)
```

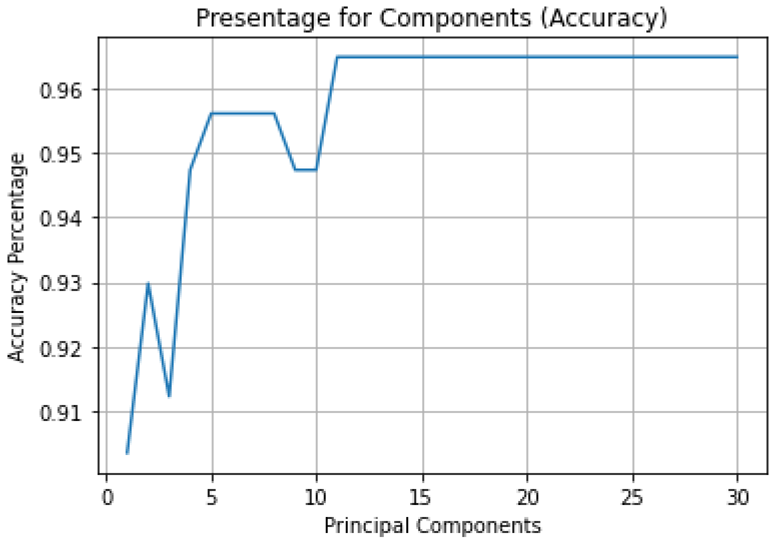
Out[405… [<matplotlib.lines.Line2D at 0x18a9c582b50>]



In [411…

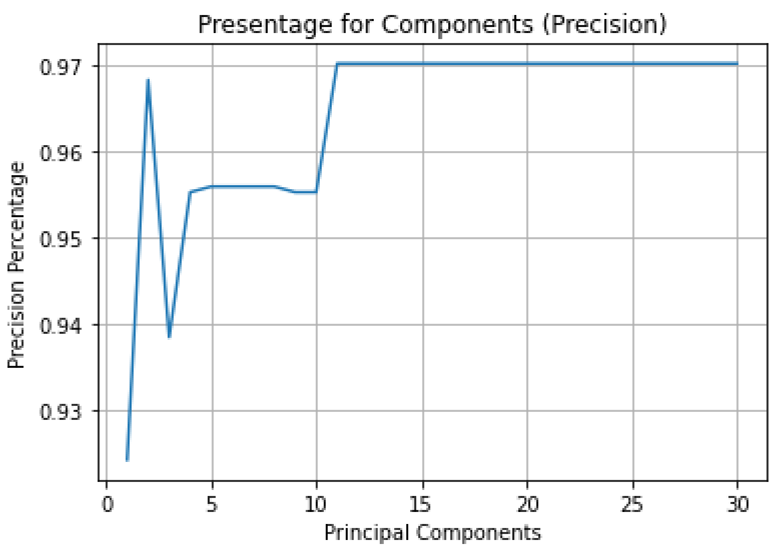
```
plt.title('Presentage for Components (Accuracy)')
plt.ylabel('Accuracy Percentage')
plt.xlabel('Principal Components')
```

```
plt.plot(K, Accuracy)
plt.grid()
```



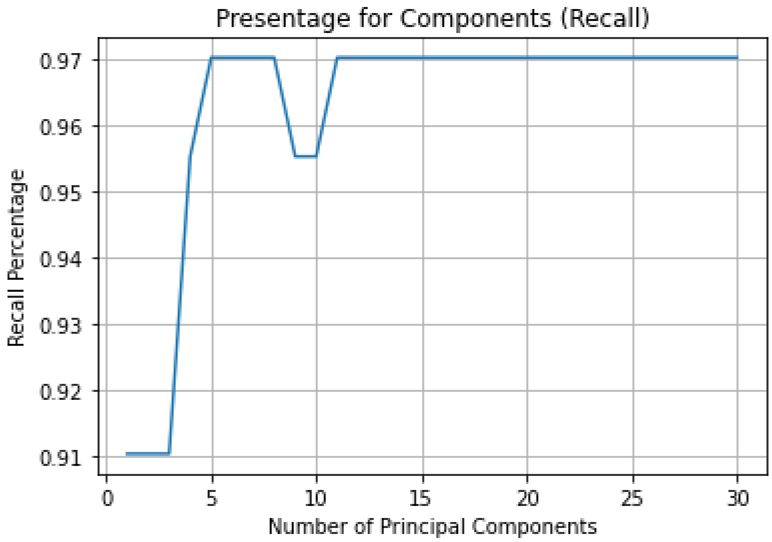
In [410…

```
plt.title('Presentage for Components (Precision)')
plt.ylabel('Precision Percentage')
plt.xlabel('Principal Components')
plt.plot(K, Precision)
plt.grid()
```



In [413…

```
plt.title('Presentage for Components (Recall)')
plt.ylabel('Recall Percentage')
plt.xlabel('Number of Principal Components')
plt.plot(K, Recall)
plt.grid()
```



In [414…

```
# Question 3: Repeat problem 2, but this time use the LDA feature extraction for your t
# For the classification, use the built-in Bays classifier for the classification.

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

X = StandardScaler().fit_transform(breast_input)

Y = breast_labels
lda = LinearDiscriminantAnalysis(n_components=1)
lda_t = lda.fit_transform(X,Y)

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2, random_state = RSTA
lda.fit(X_train,Y_train)
Y_pred = lda.predict(X_test)

from sklearn.metrics import confusion_matrix
cnf_matrix = confusion_matrix(Y_test, Y_pred)
print(cnf_matrix)

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))
```

```
[[43  4]
 [ 0 67]]
Accuracy: 0.9649122807017544
Precision: 0.9436619718309859
Recall: 1.0
```

In [415…

```
# Question 4: Can you repeat problem 3? This time, replace the Bayes classifier with lo
# Report your results (classification accuracy, precision, and recall).

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
classifier = LogisticRegression(random_state=RSTATE)
X = StandardScaler().fit_transform(breast_input)

Y = breast_labels
lda = LinearDiscriminantAnalysis(n_components=1)
lda_t = lda.fit_transform(X,Y)
classifier.fit(lda_t,Y)
```

```python
X_train,X_test,Y_train,Y_test = train_test_split(lda_t,Y,test_size=0.2, random_state =

Y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix

cnf_matrix = confusion_matrix(Y_test, Y_pred)
print(cnf_matrix)

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))
```

```
[[45  2]
 [ 1 66]]
Accuracy: 0.9736842105263158
Precision: 0.9705882352941176
Recall: 0.9850746268656716
```

In [ ]:

```

```