In [1]:
```
"""
ECGR 5105 - Intro to Machine Learning
Homework 3 - Part 2
Phillip Harmon
"""
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:
```
#Load and Build the Dataset
from sklearn.datasets import load_breast_cancer
loaded  = load_breast_cancer()
labels  = np.reshape(loaded.target, (len(loaded.target),1))
inputs  = pd.DataFrame(loaded.data)
names   = np.append(loaded.feature_names, 'label')
dataset = pd.DataFrame(np.concatenate([inputs,labels],axis=1))
dataset.columns = names
dataset
```

Out[2]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmetr
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.241
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.181
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.206
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.259
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.180
...	...	...	...	...	...	...	...	...	.
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.172
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.175
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.159
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.239
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.158

569 rows × 31 columns

In [3]:
```
#Sort Dataset
x = dataset.iloc[:,0:-1].values
y = dataset.iloc[:,-1].values
```

In [4]:
```
#Clean the Dataset
from sklearn.preprocessing import MinMaxScaler, StandardScaler
scaler = MinMaxScaler()
x = scaler.fit_transform(x)
```

In [5]:

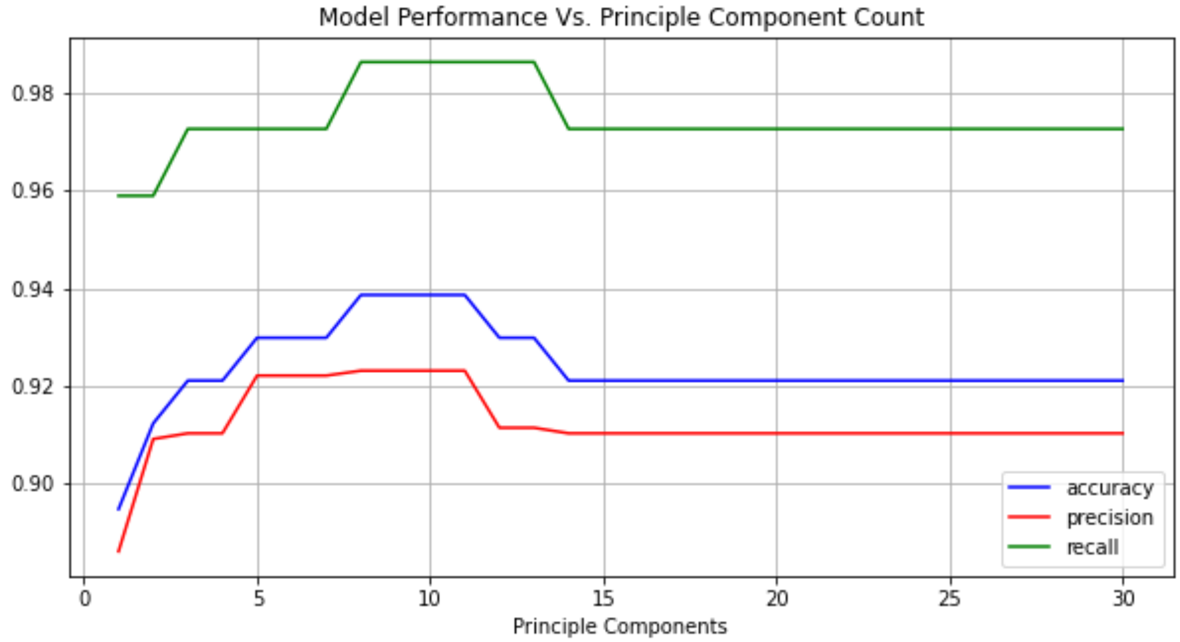
```
#Perform PCA Feature Reduction, train-test split, and train the model for a va
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, cla

frameLog = []
modelLog = []
accuracyLog = []
precisionLog = []
recallLog = []
cols = []
maxPC = len(x[0])+1

for k in range(1,maxPC):

    pca = PCA(n_components = k)
    pcs = pca.fit_transform(x)
    cols.append('PC'+str(k))
    pcFrame = pd.DataFrame(data=pcs,columns=cols)
    frameLog.append(pcFrame)

    xt, xv, yt, yv = train_test_split(pcFrame, y,
                                      train_size = 0.8, test_size = 0.2,
                                      random_state=1337)

    model = LogisticRegression(random_state=1337)
    model.fit(xt,yt);
    modelLog.append(model)

    yp = model.predict(xv)
    print("Classification Report for K={}".format(k))
    print("------------------------------------------------------")
    print(classification_report(yv,yp))
    print("Confusion Matrix")
    print(confusion_matrix(yv,yp))
    print("------------------------------------------------------\n")
    accuracyLog.append(accuracy_score(yv,yp))
    precisionLog.append(precision_score(yv,yp))
    recallLog.append(recall_score(yv,yp))
```

```
Classification Report for K=1
------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.91      0.78      0.84        41
         1.0       0.89      0.96      0.92        73

    accuracy                           0.89       114
   macro avg       0.90      0.87      0.88       114
weighted avg       0.90      0.89      0.89       114

Confusion Matrix
[[32  9]
 [ 3 70]]
------------------------------------------------------
```

```
Classification Report for K=2
------------------------------------------------------
              precision    recall  f1-score   support
```

In [6]:

```
#print the training results with a plot
plt.rcParams["figure.figsize"] = (10,5)
plt.grid()
plt.xlabel('Principle Components')
plt.title('Model Performance Vs. Principle Component Count')
plt.plot(range(1,maxPC),accuracyLog,color='blue',label='accuracy')
plt.plot(range(1,maxPC),precisionLog,color='red',label='precision')
plt.plot(range(1,maxPC),recallLog,color='green',label='recall')
plt.legend();
```

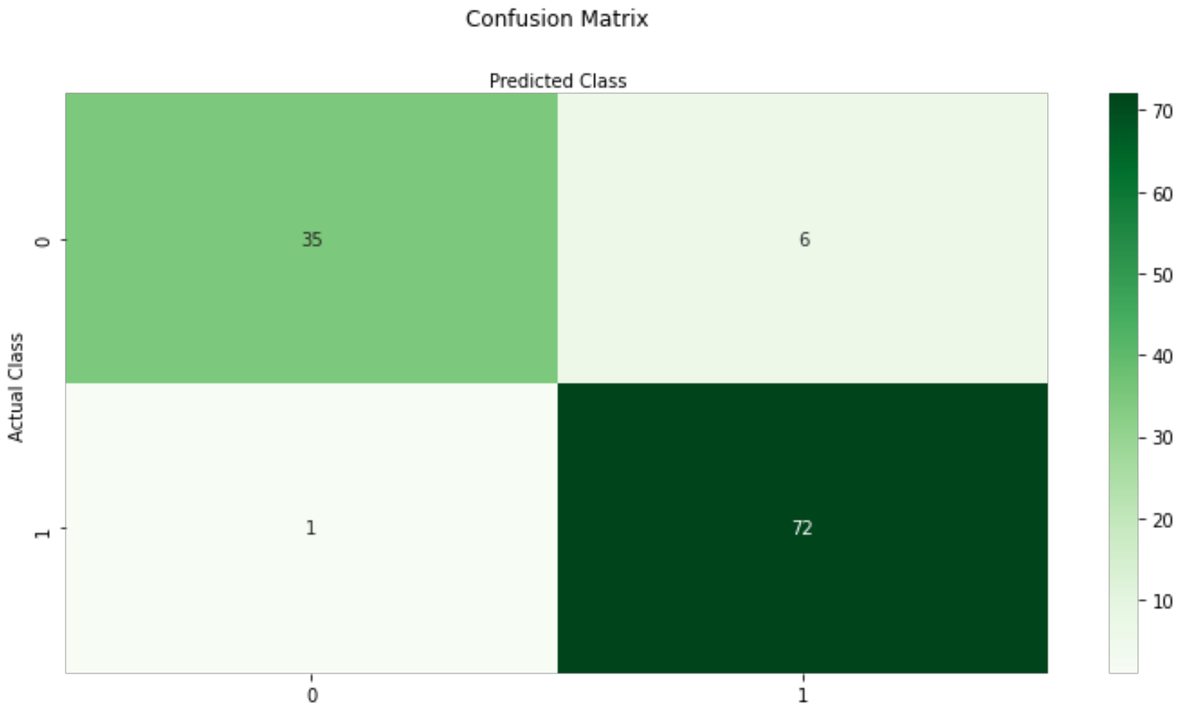


In [7]:

```python
#Print Best Results
K = accuracyLog.index(max(accuracyLog))
print("According to the plot above, the highest accuracy occurs at a lowest di
xt, xv, yt, yv = train_test_split(frameLog[K], y,train_size = 0.8, test_size =
yp = modelLog[K].predict(xv)
print("Classification Report for K={}".format(K+1))
print("------------------------------------------------------")
print(classification_report(yv,yp))
```

```
According to the plot above, the highest accuracy occurs at a lowest dimensio
nality of K=8


Classification Report for K=8
------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.97      0.85      0.91        41
         1.0       0.92      0.99      0.95        73

    accuracy                           0.94       114
   macro avg       0.95      0.92      0.93       114
weighted avg       0.94      0.94      0.94       114
```

```
In [8]: #Analyze using the Confusion Matrix
        import seaborn as sns
        classes = ['Benign','Malignant']
        figure, axis = plt.subplots()
        ticks = np.arange(len(classes))
        plt.xticks(ticks, classes)
        plt.yticks(ticks, classes)
        sns.heatmap(pd.DataFrame(confusion_matrix(yv,yp)),
                    annot=True, cmap="Greens", fmt='g')
        axis.xaxis.set_label_position("top")
        plt.tight_layout()
        plt.title('Confusion Matrix', y=1.1)
        plt.ylabel('Actual Class')
        plt.xlabel('Predicted Class');
```



```
In [ ]:
```