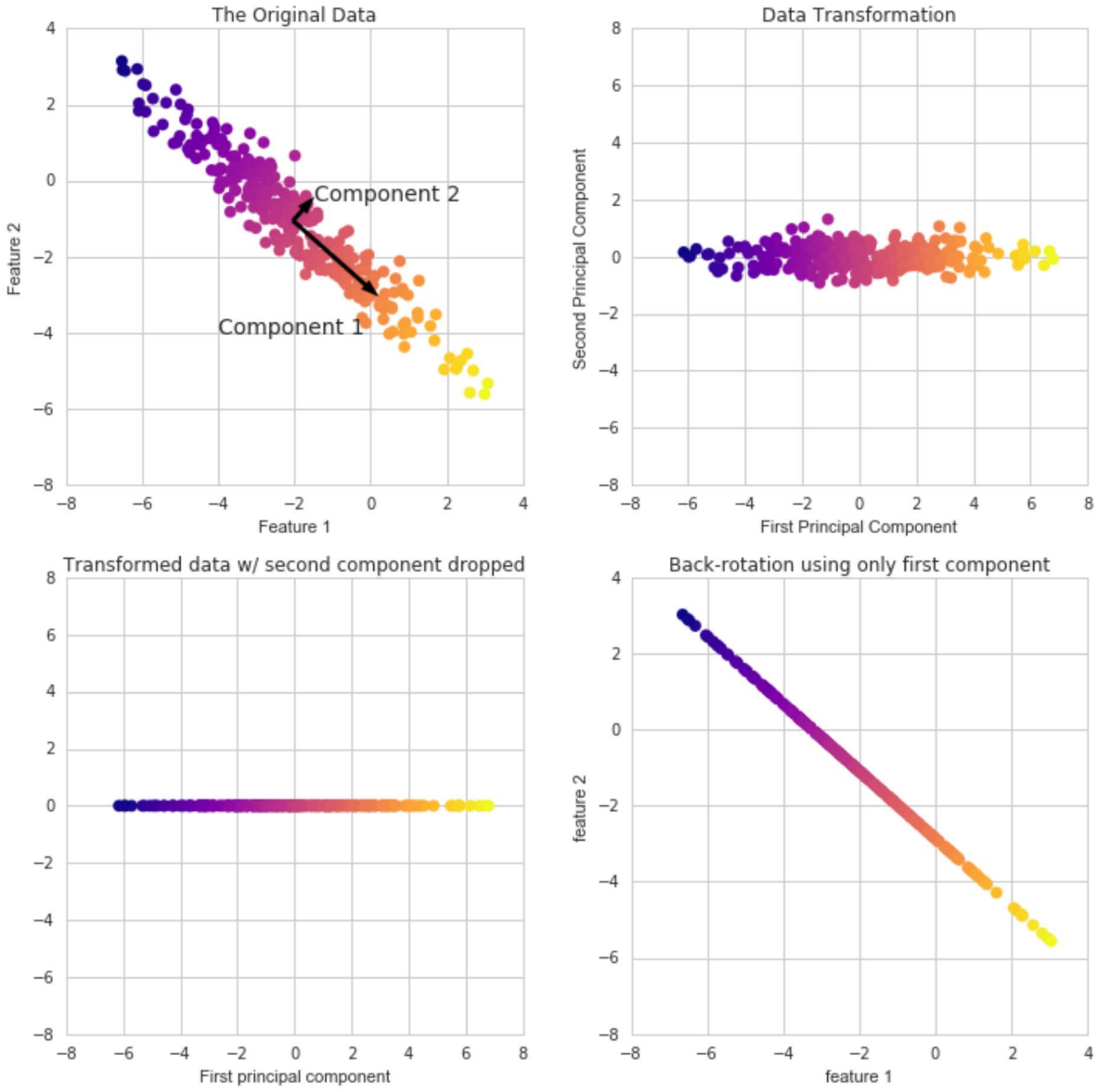
Principal Component Analysis

PCA isn't exactly a full machine learning algorithm, but instead an unsupervised learning algorithm

The key aim of PCA is to reduce the number of variables of a data set, while preserving as much information as possible.

PCA Review



Advantages by using PCA:

- Removes correlated features. PCA will help you remove all the features that are correlated, a phenomenon known as multi-collinearity. Finding features that are correlated is time consuming, especially if the number of features is large.

- Improves machine learning algorithm performance. With the number of features reduced with PCA, the time taken to train your model is now significantly reduced.
- Reduce overfitting. By removing the unnecessary features in your dataset, PCA helps to overcome overfitting.

Disadvantages by using PCA:

- Independent variables are now less interpretable. PCA reduces your features into smaller number of components. Each component is now a linear combination of your original features, which makes it less readable and interpretable.
- Information loss. Data loss may occur if you do not exercise care in choosing the right number of components.
- Feature scaling. Because PCA is a variance maximizing exercise, PCA requires features to be scaled prior to processing.

Libraries

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

The Data

Let's work with the cancer data set again since it had so many features.

In [2]:
```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
```

In [3]:
```
cancer.keys()
```

Out[3]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'file
name', 'data_module'])
```

In [4]:
```
print(cancer['DESCR'])
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the three
        worst/largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 0 is Mean Radius, field
        10 is Radius SE, field 20 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign

    :Summary Statistics:

    ===================================== ====== ======
                                           Min    Max
    ===================================== ====== ======
    radius (mean):                        6.981  28.11
    texture (mean):                       9.71   39.28
    perimeter (mean):                     43.79  188.5
    area (mean):                          143.5  2501.0
    smoothness (mean):                    0.053  0.163
    compactness (mean):                   0.019  0.345
    concavity (mean):                     0.0    0.427
    concave points (mean):                0.0    0.201
    symmetry (mean):                      0.106  0.304
    fractal dimension (mean):             0.05   0.097
    radius (standard error):              0.112  2.873
    texture (standard error):             0.36   4.885
    perimeter (standard error):           0.757  21.98
    area (standard error):                6.802  542.2
    smoothness (standard error):          0.002  0.031
    compactness (standard error):         0.002  0.135
    concavity (standard error):           0.0    0.396
    concave points (standard error):      0.0    0.053
    symmetry (standard error):            0.008  0.079
    fractal dimension (standard error):   0.001  0.03
    radius (worst):                       7.93   36.04
    texture (worst):                      12.02  49.54
    perimeter (worst):                    50.41  251.2
```

```
    area (worst):                          185.2  4254.0
    smoothness (worst):                    0.071  0.223
    compactness (worst):                   0.027  1.058
    concavity (worst):                     0.0    1.252
    concave points (worst):                0.0    0.291
    symmetry (worst):                      0.156  0.664
    fractal dimension (worst):             0.055  0.208
    ====================================== ====== ======

    :Missing Attribute Values: None

    :Class Distribution: 212 - Malignant, 357 - Benign

    :Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

    :Donor: Nick Street

    :Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References

   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
     for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
     Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
     San Jose, CA, 1993.
   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
     prognosis via linear programming. Operations Research, 43(4), pages 570-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994)
     163-171.
```

In [5]:
```
df = pd.DataFrame(cancer['data'],columns=cancer['feature_names'])
```

```
#(['DESCR', 'data', 'feature_names', 'target_names', 'target'])
df.head()
```

Out[5]:

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

5 rows × 30 columns

PCA Visualization

As we've noticed before it is difficult to visualize high dimensional data, we can use PCA to find the first two principal components, and visualize the data in this new, two-dimensional space, with a single scatter-plot. Before we do this though, we'll need to scale our data so that each feature has a single unit variance.

In [7]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df)
```

Out[7]:
```
StandardScaler
StandardScaler()
```

In [8]:
```
scaled_data = scaler.transform(df)
```

PCA with Scikit Learn uses a very similar process to other preprocessing functions that come with SciKit Learn. We instantiate a PCA object, find the principal components using the fit method, then apply the rotation and dimensionality reduction by calling transform().

We can also specify how many components we want to keep when creating the PCA object.

In [13]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(scaled_data)
```

Out[13]:
```
PCA
PCA(n_components=2)
```

Now we can transform this data to its first 2 principal components.

In [14]:
```
x_pca = pca.transform(scaled_data)
```

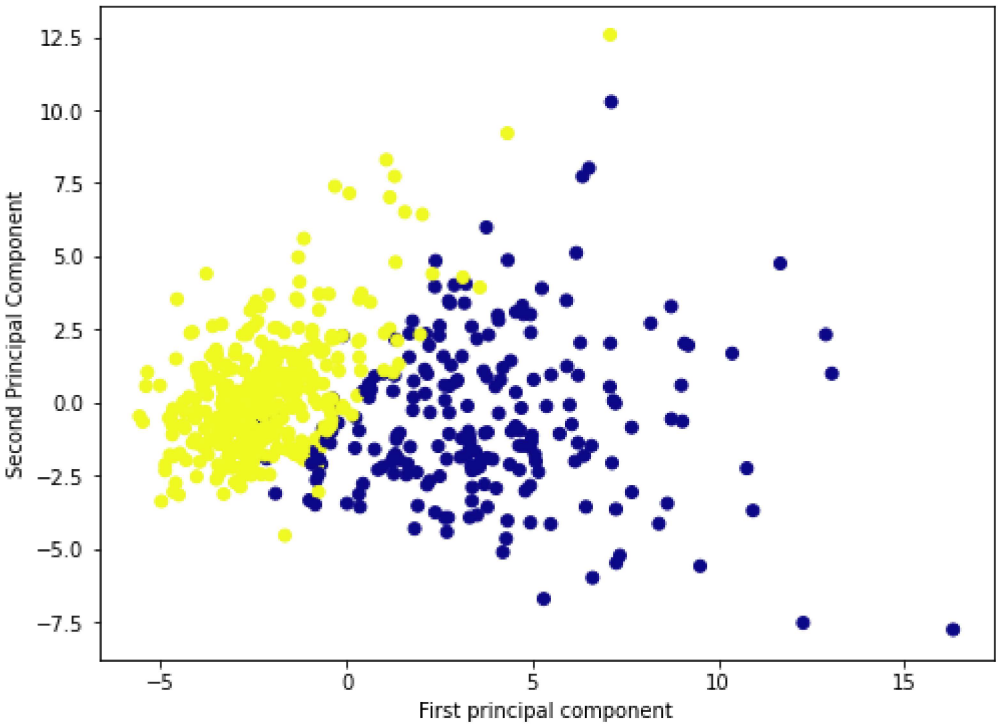
In [16]:
```
scaled_data.shape,x_pca.shape
```

Out[16]:
```
((569, 30), (569, 2))
```

We've reduced 30 dimensions to just 2! Let's plot these two dimensions out!

In [17]:
```
plt.figure(figsize=(8,6))
plt.scatter(x_pca[:,0],x_pca[:,1],c=cancer['target'],cmap='plasma')
plt.xlabel('First principal component')
plt.ylabel('Second Principal Component')
```

Out[17]:
```
Text(0, 0.5, 'Second Principal Component')
```



Clearly by using these two components we can easily separate these two classes.

Interpreting the components

Unfortunately, with this great power of dimensionality reduction, comes the cost of being able to easily understand what these components represent.

The components correspond to combinations of the original features, the components themselves are stored as an attribute of the fitted PCA object:

In [18]:
```
pca.components_
```

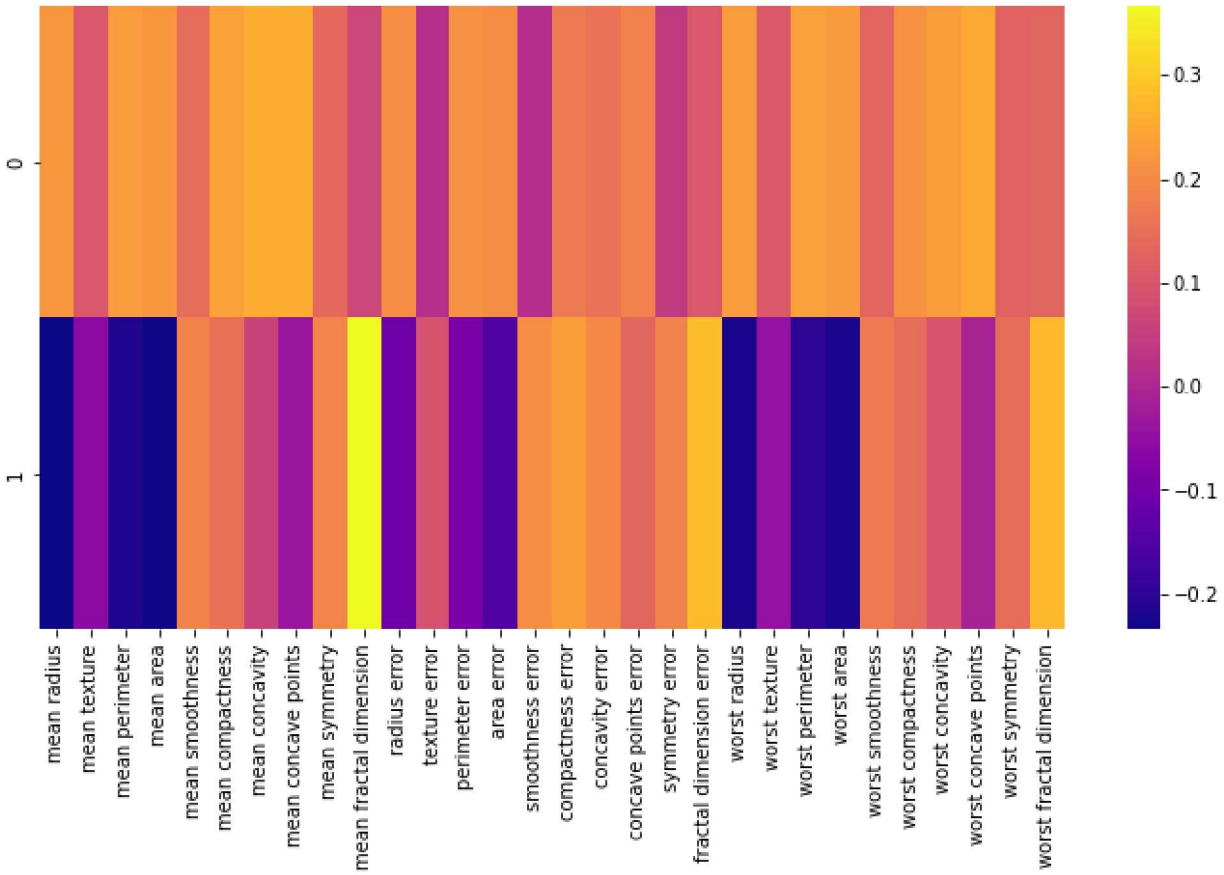
Out[18]:
```
array([[ 0.21890244,  0.10372458,  0.22753729,  0.22099499,  0.14258969,
         0.23928535,  0.25840048,  0.26085376,  0.13816696,  0.06436335,
         0.20597878,  0.01742803,  0.21132592,  0.20286964,  0.01453145,
         0.17039345,  0.15358979,  0.1834174 ,  0.04249842,  0.10256832,
         0.22799663,  0.10446933,  0.23663968,  0.22487053,  0.12795256,
         0.21009588,  0.22876753,  0.25088597,  0.12290456,  0.13178394],
       [-0.23385713, -0.05970609, -0.21518136, -0.23107671,  0.18611302,
         0.15189161,  0.06016536, -0.0347675 ,  0.19034877,  0.36657547,
        -0.10555215,  0.08997968, -0.08945723, -0.15229263,  0.20443045,
         0.2327159 ,  0.19720728,  0.13032156,  0.183848  ,  0.28009203,
        -0.21986638, -0.0454673 , -0.19987843, -0.21935186,  0.17230435,
         0.14359317,  0.09796411, -0.00825724,  0.14188335,  0.27533947]])
```

In this numpy matrix array, each row represents a principal component, and each column relates back to the original features. we can visualize this relationship with a heatmap:

In [19]:
```
df_comp = pd.DataFrame(pca.components_,columns=cancer['feature_names'])
```

In [20]:
```
plt.figure(figsize=(12,6))
sns.heatmap(df_comp,cmap='plasma',)
```

Out[20]:
```
<AxesSubplot:>
```



This heatmap and the color bar basically represent the correlation between the various feature and the principal component itself.

In [ ]: