Objective: To understand the importance of scaling on PCA

In [8]:
```
from sklearn.decomposition import PCA
from sklearn import preprocessing
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
```

Task 0: Write the function to compute the pca using Eigenvector approach

In [9]:
```
from numpy.linalg import svd
def pca(X):
    U, S, P_trans = svd(X, full_matrices = False)
    Sigma = np.diag(S)
    T = np.dot(U,Sigma)
    P = P_trans.T
    return T, Sigma, P #Score, Variace, Loadings
```

In [10]:
```
features, target = load_wine(return_X_y=True)
X=features
y=target
```

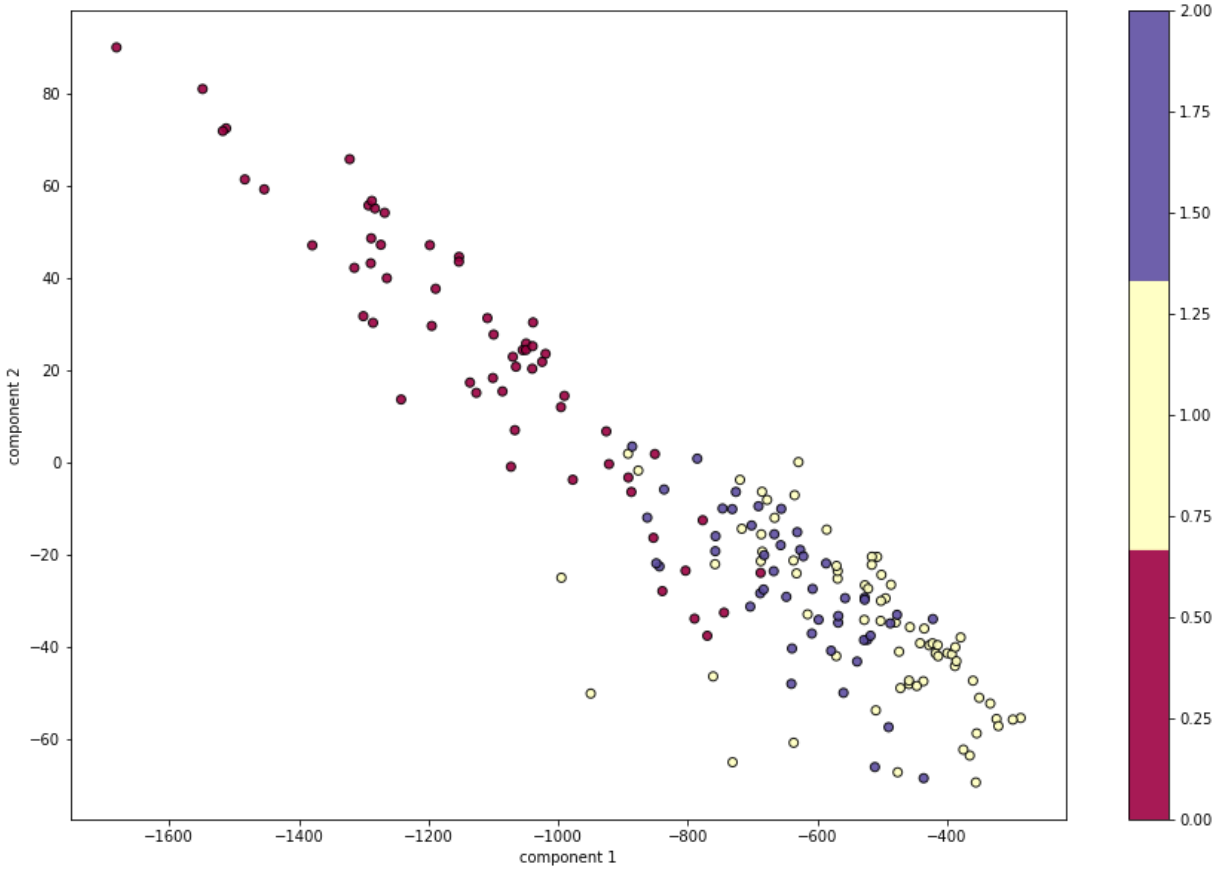
Three different ways of scaling

- Scaling by removing the mean and divising by the standard deviation
  `#standard_scaling=preprocessing.StandardScaler()`
  `#X_standard=standard_scaling.fit_transform(X)`
- Scaling to min and maximum values of each feature
  `#minmax_scaling=preprocessing.MinMaxScaler()`
  `#X_minmax=minmax_scaling.fit_transform(X)`
- Scaling by diving by the maximumabsolute values of each features
  `#max_abs_scaler=preprocessing.MaxAbsScaler()`
  `#X_maxabs=max_abs_scaler.fit_transform(X)`

Task 1: Create the scores plot without any scaling

In [13]:
```
T,S,P=pca(X)
plt.figure(figsize=(15,10))
print(T.shape)
plt.scatter(T[:, 0], T[:, 1],
            c=y, edgecolor = 'black', alpha=0.9,
            cmap=plt.cm.get_cmap('Spectral', 3))
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.colorbar();
```

```
(178, 13)
```



Task 2: Create the scores plot with standard scaling

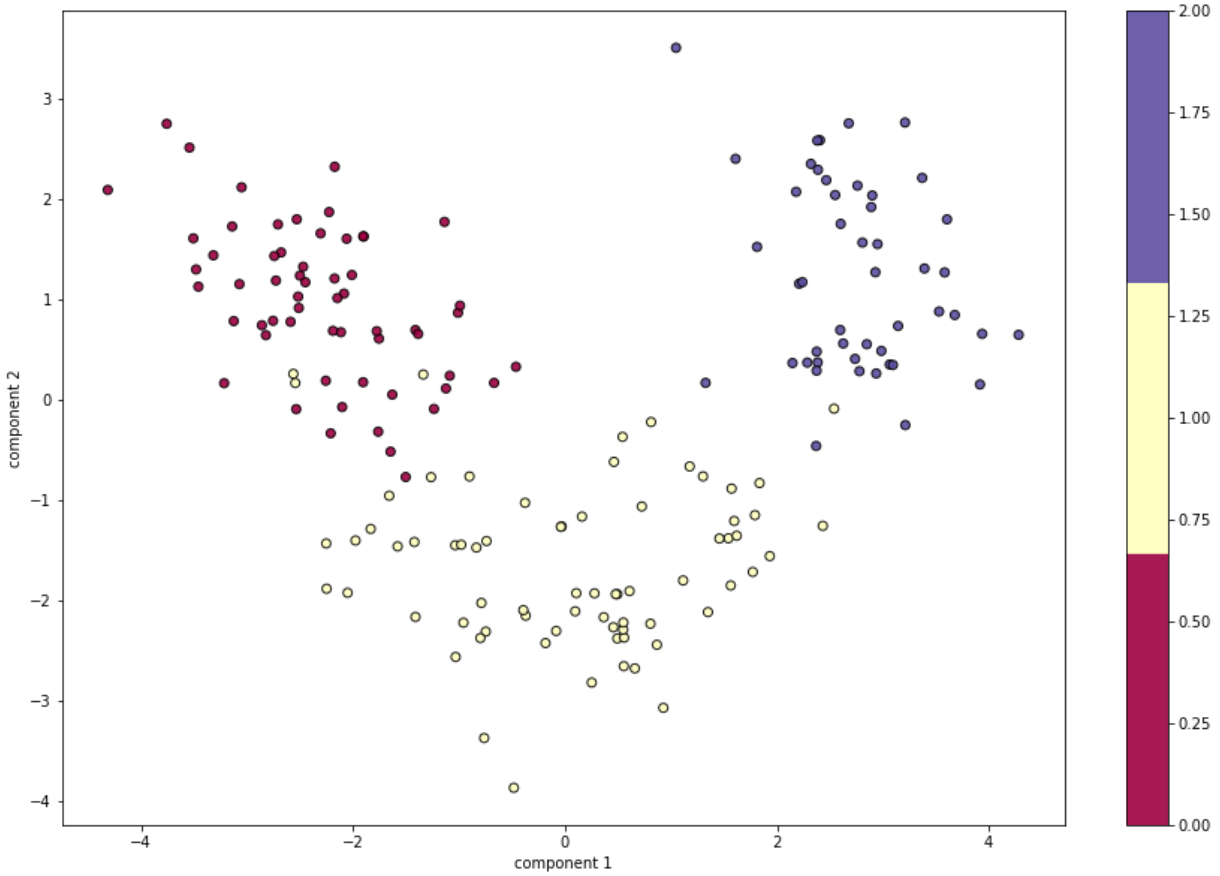
In [16]:
```
# Standard scaling removes mean and scales to unit variance. I.e. out = (x - mean) /

standard_scaling=preprocessing.StandardScaler()
X_standard=standard_scaling.fit_transform(X)

T,S,P=pca(X_standard)

plt.figure(figsize=(15,10))
print(T.shape)
plt.scatter(T[:, 0], T[:, 1],
            c=y, edgecolor='black', alpha=0.9,
            cmap=plt.cm.get_cmap('Spectral', 3))
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.colorbar();
```

```
(178, 13)
```

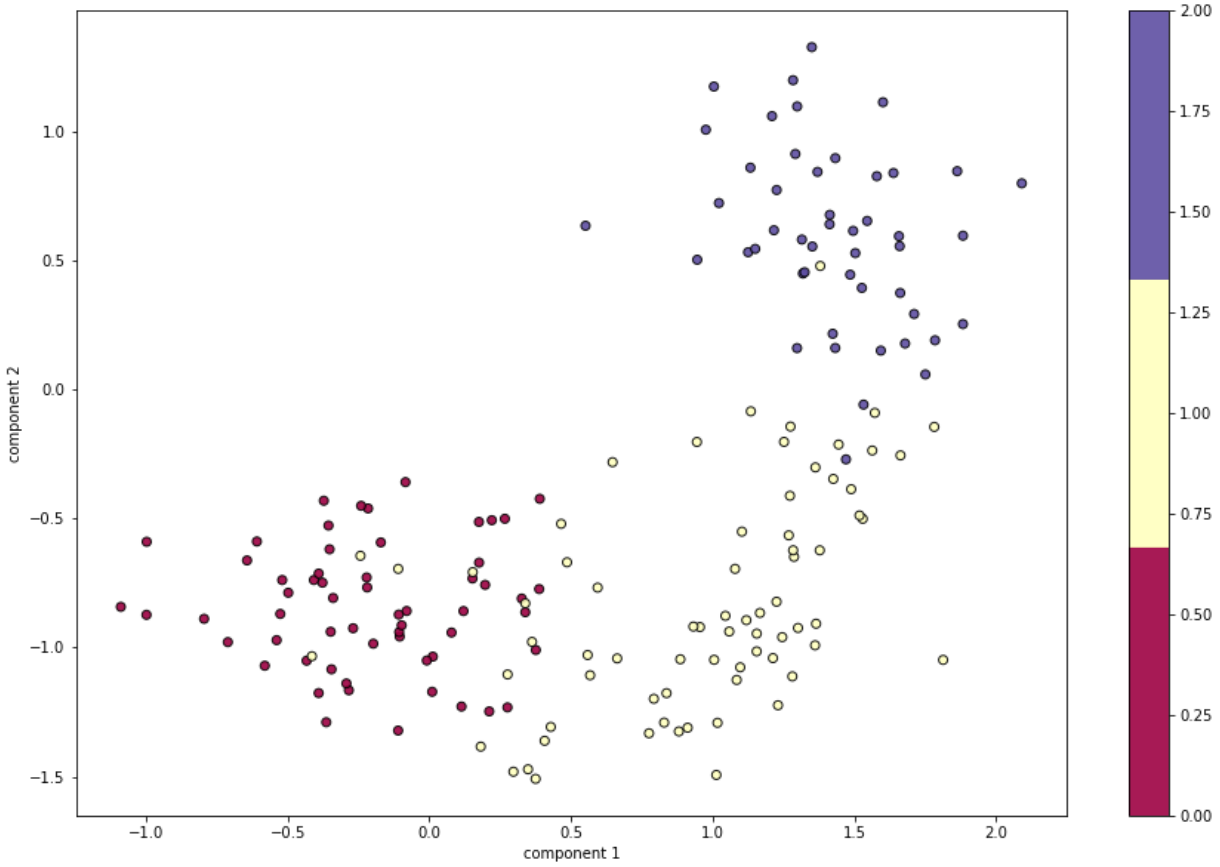


Task 3: Create the scores plot with min max scaling

In [25]:
```
# Min Max scaling transformes the features to fit within a given range [min, max]. T
# X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
# X_scaled = X_std * (max - min) + min
print(np.max(X))
minmax_scaling=preprocessing.MinMaxScaler((-1,1))
X_minmax=minmax_scaling.fit_transform(X)
T,S,P=pca(X_minmax)

plt.figure(figsize=(15,10))
plt.scatter(T[:, 0], T[:, 1],
            c=y, edgecolor='black', alpha=0.9,
            cmap=plt.cm.get_cmap('Spectral', 3))
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.colorbar();
```

```
1680.0
```



Task 4: Create the scores plot with max abs scaling

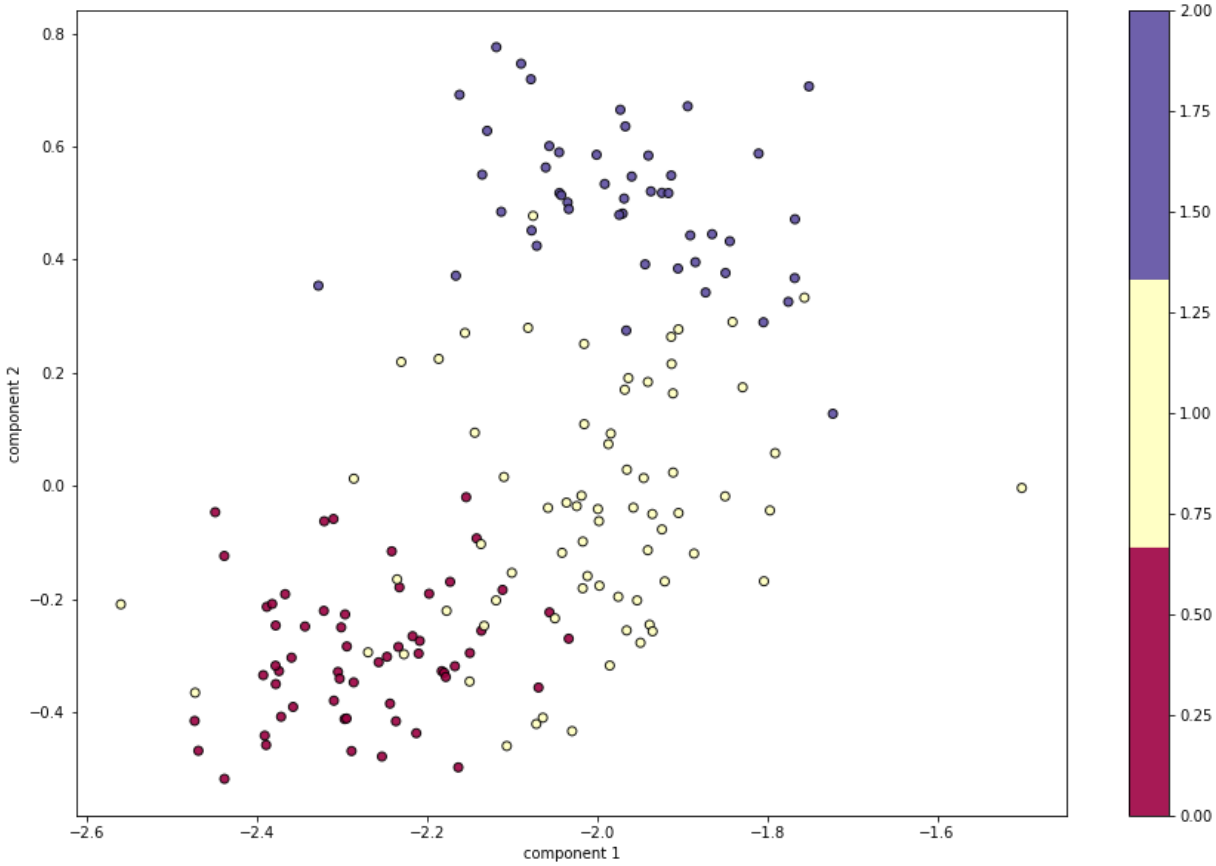
In [26]:
```
# Max abs scaling scales each feature so that they have maximal absolute value of 1.

max_abs_scaler=preprocessing.MaxAbsScaler()
X_maxabs=max_abs_scaler.fit_transform(X)

T,S,P=pca(X_maxabs)

plt.figure(figsize=(15,10))
print(T.shape)
plt.scatter(T[:, 0], T[:, 1],
            c=y, edgecolor='black', alpha=0.9,
            cmap=plt.cm.get_cmap('Spectral', 3))
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.colorbar();
```

```
(178, 13)
```


component 2
component 1
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−2.6
−2.4
−2.2
−2.0
−1.8
−1.6
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

In [ ]: