In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
from sklearn.datasets import load_breast_cancer
```

In [3]:

```
cancer_data = load_breast_cancer()
```

In [4]:

```
cancer_data.keys()
```

Out[4]:

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names'])
```

In [5]:

```
len(cancer_data['feature_names'])
```

Out[5]:

```
30
```

In [7]:

```
df = pd.DataFrame(cancer_data['data'], columns=cancer_data['feature_names'])
```

In [8]:

```
df.head()
```

Out[8]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	v perin
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	18
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	15
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	15
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	9
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	15

5 rows × 30 columns

In [9]:

```
from sklearn.preprocessing import StandardScaler
```

In [10]:

```
scaler = StandardScaler()
```

In [11]:

```
scaler.fit(df)
```

Out[11]:

```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [12]:

```
df_scaled = scaler.transform(df)
```

In [16]:

```
from sklearn.decomposition import PCA
```

In [17]:

```
pca = PCA(n_components=2)
```

In [18]:

```
pca.fit(df_scaled)
```

Out[18]:

```
PCA(copy=True, iterated_power='auto', n_components=2, random_state=None,
  svd_solver='auto', tol=0.0, whiten=False)
```

In [19]:

```
pc12 = pca.transform(df_scaled)
```

In [20]:

```
df_pca = pd.DataFrame(pc12, columns=['PC1', 'PC2'])
```

In [21]:

```
df_pca.head()
```

Out[21]:

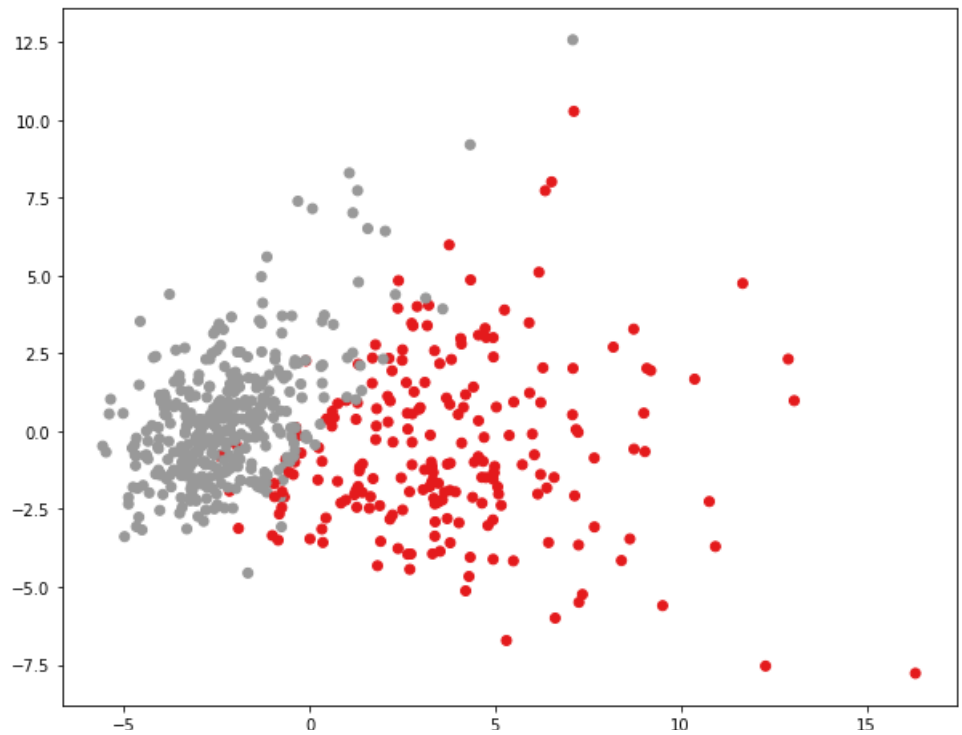
	PC1	PC2
0	9.192837	1.948583
1	2.387802	-3.768172
2	5.733896	-1.075174
3	7.122953	10.275589
4	3.935302	-1.948072

In [37]:

```
plt.figure(figsize=(10, 8))
plt.scatter(x=df_pca['PC1'], y=df_pca['PC2'], c=cancer_data['target'], cmap='Set1')
```

Out[37]:

```
<matplotlib.collections.PathCollection at 0x1a24c4da20>
```



In [38]:

```
pca.components_
```

Out[38]:

```
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 [39]:

```
df_pca_comp = pd.DataFrame(pca.components_, columns=cancer_data['feature_names'])
```

In [40]:

```
df_pca_comp.head()
```

Out[40]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	te
0	0.218902	0.103725	0.227537	0.220995	0.142590	0.239285	0.258400	0.260854	0.138167	0.064363	...	0.227997	0.1(
1	-0.233857	-0.059706	-0.215181	-0.231077	0.186113	0.151892	0.060165	-0.034768	0.190349	0.366575	...	-0.219866	0.0

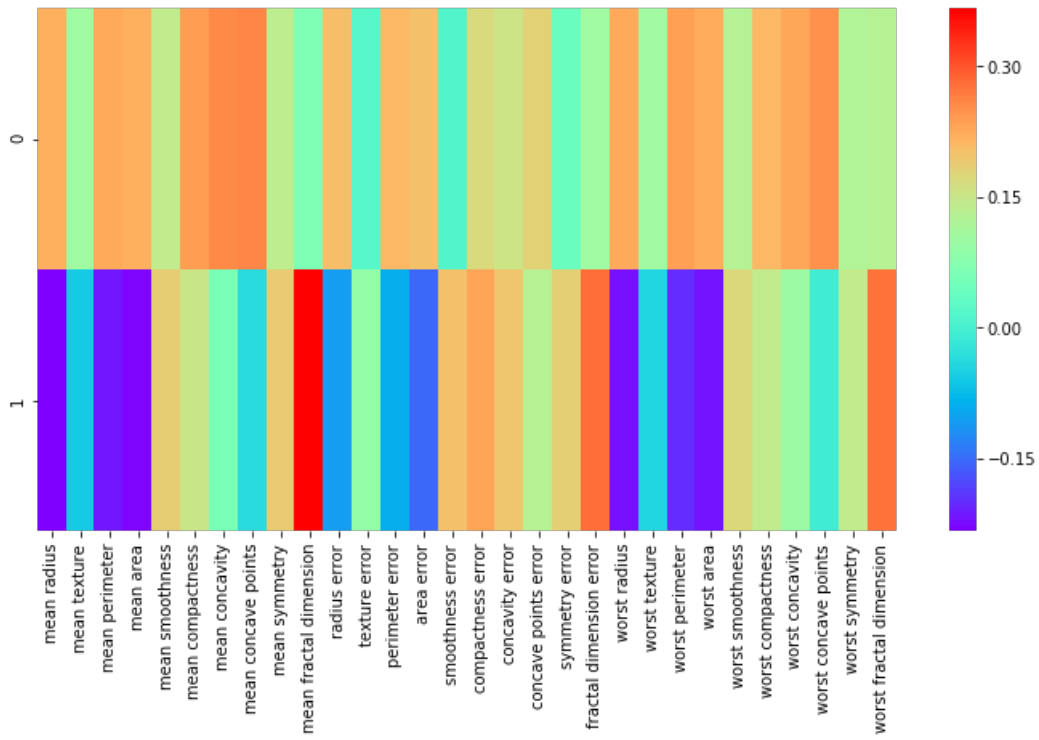
2 rows × 30 columns

In [41]:

```
plt.figure(figsize=(12, 6))
sns.heatmap(df_pca_comp, cmap='rainbow')
```

Out[41]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2479b320>
```



In [42]:

```
x = df_pca
y = cancer_data['target']
```

In [43]:

```
from sklearn.model_selection import train_test_split
```

In [44]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

In [45]:

```
from sklearn.linear_model import LogisticRegression
```

In [46]:

```
logR = LogisticRegression()
```

In [47]:

```
logR.fit(x_train, y_train)
```

Out[47]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [48]:

```
predictions = logR.predict(x_test)
```

In [49]:

```
from sklearn.metrics import confusion_matrix, classification_report
```

In [50]:

```
print(classification_report(y_test, predictions))
```

```
             precision    recall  f1-score   support

          0       0.97      0.95      0.96        63
          1       0.97      0.98      0.98       108

avg / total       0.97      0.97      0.97       171
```

In [51]:

```
print(confusion_matrix(y_test, predictions))
```

```
[[ 60   3]
 [  2 106]]
```

In [ ]:

```

```