演習：主成分分析（sklearn）

- 設定
  - 乳がん検査データを利用しロジスティック回帰モデルを作成
  - 主成分を利用し2次元空間上に次元圧縮
- 課題
  - 32次元のデータを2次元上に次元圧縮した際に、うまく判別できるかを確認

In [26]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegressionCV
from sklearn.metrics import confusion_matrix
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('ggplot')
```

In [2]:

```
cancer_df = pd.read_csv('./data/cancer.csv', encoding='utf-8')
```

データの確認

In [7]:

```
print(cancer_df.shape)
display(cancer_df.head())
```

(569, 33)

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothnes
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

5 rows × 33 columns

前処理

In [8]:

```
cancer_df = cancer_df.drop('Unnamed: 32', axis=1)  # 不要列を削除
```

In [9]:

```
print(cancer_df.shape)
```

(569, 32)

- diagnosis: 診断結果 (良性がB / 悪性がM)
- 説明変数は3列以降、目的変数を2列目としロジスティック回帰で分類

In [10]:

```
# 診断結果を {良性: 0, 悪性: 1} に変換
cancer_df['diagnosis'] = cancer_df['diagnosis'].apply(lambda x: 1 if x == 'M' else 0)
```

In [11]:

```
# 目的変数の抽出
y = cancer_df['diagnosis']
```

In [12]:

```
# 説明変数の抽出
X = cancer_df.drop('diagnosis', axis=1)
```

In [13]:

```
display(X.head(3))
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	cc
0	842302	17.99	10.38	122.8	1001.0	0.11840	
1	842517	20.57	17.77	132.9	1326.0	0.08474	
2	84300903	19.69	21.25	130.0	1203.0	0.10960	

3 rows × 31 columns

学習

In [14]:

```
# 学習用とテスト用でデータを分離
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

In [15]:

```
# 標準化
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [16]:

```
# ロジスティック回帰で学習
model_lr = LogisticRegressionCV(cv=10, random_state=0)
```

モデルの検証

In [18]:

```
# スコアの確認
score_train = model_lr.score(X_train_scaled, y_train)
score_test = model_lr.score(X_test_scaled, y_test)

print(f'Train score: {score_train:.3f}')
print(f'Test score: {score_test:.3f}')
```

```
Train score: 0.988
Test score: 0.965
```

In [19]:

```
# テストデータから予測
y_pred = model_lr.predict(X_test_scaled)
```

In [21]:

```
# 混同行列
confusion_mat = confusion_matrix(y_true=y_test, y_pred=y_pred)
print(f'Confustion Matrix:\n{confusion_mat}')
```

```
Confustion Matrix:
[[89  1]
 [ 4 49]]
```

考察：
ロジスティック回帰では、分類精度自体は非常に高い結果となった。
ただし、学習データのスコアよりテストデータのスコアが悪いことから過学習が疑われる。

主成分分析

In [36]:

```
pca = PCA(n_components=30)
pca.fit(X_train_scaled)
```

Out[36]:

```
PCA(n_components=30)
```

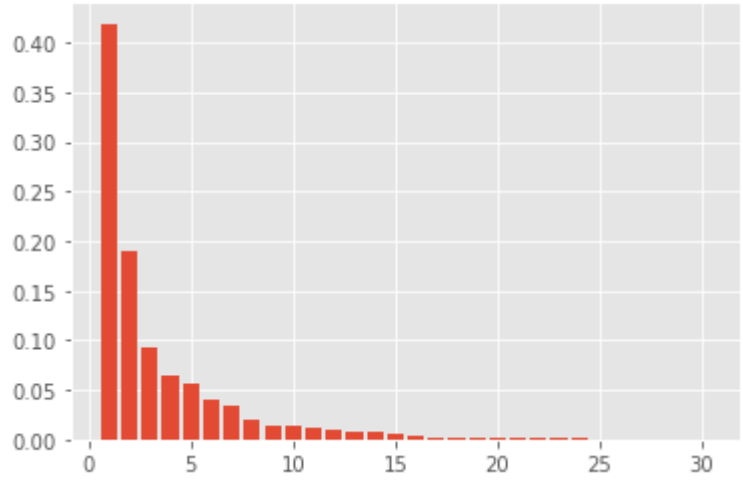
In [37]:

```
# 寄与率
ratio = pca.explained_variance_ratio_
print(ratio)
print(len(ratio))
```

```
[4.19356639e-01 1.89758038e-01 9.28752931e-02 6.47347758e-02
 5.53789199e-02 3.92849216e-02 3.30951990e-02 1.98419580e-02
 1.44186823e-02 1.30499234e-02 1.12985749e-02 1.00544239e-02
 8.63128818e-03 7.60047785e-03 4.85373978e-03 3.03015543e-03
 2.56268889e-03 1.92464885e-03 1.59237279e-03 1.44838730e-03
 1.02528778e-03 1.01315561e-03 8.26568340e-04 6.97877320e-04
 5.81699935e-04 5.28997538e-04 2.51858423e-04 2.09625384e-04
 4.54459005e-05 2.48265533e-05]
30
```

In [38]:

```
# 寄与率をグラフ化
plt.bar([n for n in range(1, len(ratio) + 1)], ratio)
plt.show()
```



考察：
寄与率を見ると、最初の2成分は非常に重要に見える。
7成分くらいまでは、必要そうに見える。

次元圧縮

In [40]:

```
# 2次元に次元圧縮する
pca2 = PCA(n_components=2)
X_train_pca2 = pca2.fit_transform(X_train_scaled)
```

In [41]:

```
print(X_train_pca2.shape)
```

```
(426, 2)
```

In [43]:

```
# 寄与率 (リストの要素数が2であることを確認)
ratio2 = pca2.explained_variance_ratio_
print(ratio2)
print(len(ratio2))
```

```
[0.41935664 0.18975804]
2
```

In [45]:

```
tmp_df = pd.DataFrame(X_train_pca2)
display(tmp_df.head())
```

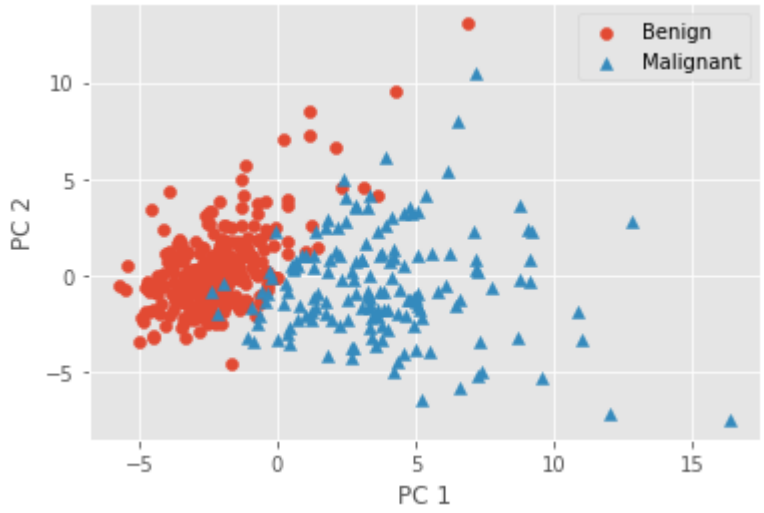
	0	1
0	-2.863510	-0.280603
1	-3.267414	1.073180
2	3.752043	-3.397107
3	-3.493976	-2.684386
4	-0.747713	-2.469012

In [47]:

```
tmp_df['Outcome'] = y_train.values
benign = tmp_df[tmp_df['Outcome'] == 0]  # 0が両性
malignant = tmp_df[tmp_df['Outcome'] == 1]  # 1が悪性
```

In [49]:

```
plt.scatter(x=benign[0], y=benign[1], marker='o', label='Benign') # 良性は○でマーク
plt.scatter(x=malignant[0], y=malignant[1], marker='^', label='Malignant') # 悪性は△でマーク
plt.xlabel('PC 1') # 第1主成分をx軸
plt.ylabel('PC 2') # 第2主成分をy軸
plt.legend()
plt.show()
```



考察：
次元圧縮後のクラスタ分類をした結果、ある直線を境界に分類されたように見える。
良性が悪性に分類されたように見える部分があるが、その逆はあまり無さそうに見える。
良性か悪性かの分類では、良性が悪性と分類される分（偽陽性）には問題は少ないと思われるため、
中々良いクラスタ分類が出来ているようには見える。