2_PCA

July 6, 2025

Creado por:

Isabel Maniega

1 PCA

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

[2]:
```
dataset = pd.read_csv("Wine.csv")
dataset
```

[2]:
```
     Alcohol  Malic_Acid   Ash  Ash_Alcanity  Magnesium  Total_Phenols  \
0      14.23        1.71  2.43          15.6        127           2.80
1      13.20        1.78  2.14          11.2        100           2.65
2      13.16        2.36  2.67          18.6        101           2.80
3      14.37        1.95  2.50          16.8        113           3.85
4      13.24        2.59  2.87          21.0        118           2.80
..       …           …     …             …          …             …
173    13.71        5.65  2.45          20.5         95           1.68
174    13.40        3.91  2.48          23.0        102           1.80
175    13.27        4.28  2.26          20.0        120           1.59
176    13.17        2.59  2.37          20.0        120           1.65
177    14.13        4.10  2.74          24.5         96           2.05

     Flavanoids  Nonflavanoid_Phenols  Proanthocyanins  Color_Intensity   Hue  \
0          3.06                  0.28             2.29             5.64  1.04
1          2.76                  0.26             1.28             4.38  1.05
2          3.24                  0.30             2.81             5.68  1.03
3          3.49                  0.24             2.18             7.80  0.86
4          2.69                  0.39             1.82             4.32  1.04
..          …                     …                …                …     …
173        0.61                  0.52             1.06             7.70  0.64
174        0.75                  0.43             1.41             7.30  0.70
175        0.69                  0.43             1.35            10.20  0.59
```

```
176        0.68                  0.53            1.46            9.30  0.60
177        0.76                  0.56            1.35            9.20  0.61

     OD280  Proline  Customer_Segment
0     3.92     1065                 1
1     3.40     1050                 1
2     3.17     1185                 1
3     3.45     1480                 1
4     2.93      735                 1
..     …        …                …
173   1.74      740                 3
174   1.56      750                 3
175   1.56      835                 3
176   1.62      840                 3
177   1.60      560                 3

[178 rows x 14 columns]
```

[3]:
```
dataset.shape
```

[3]:
```
(178, 14)
```

[4]:
```
X = dataset.iloc[:, 0:13].values
y = dataset.iloc[:, 13].values
```

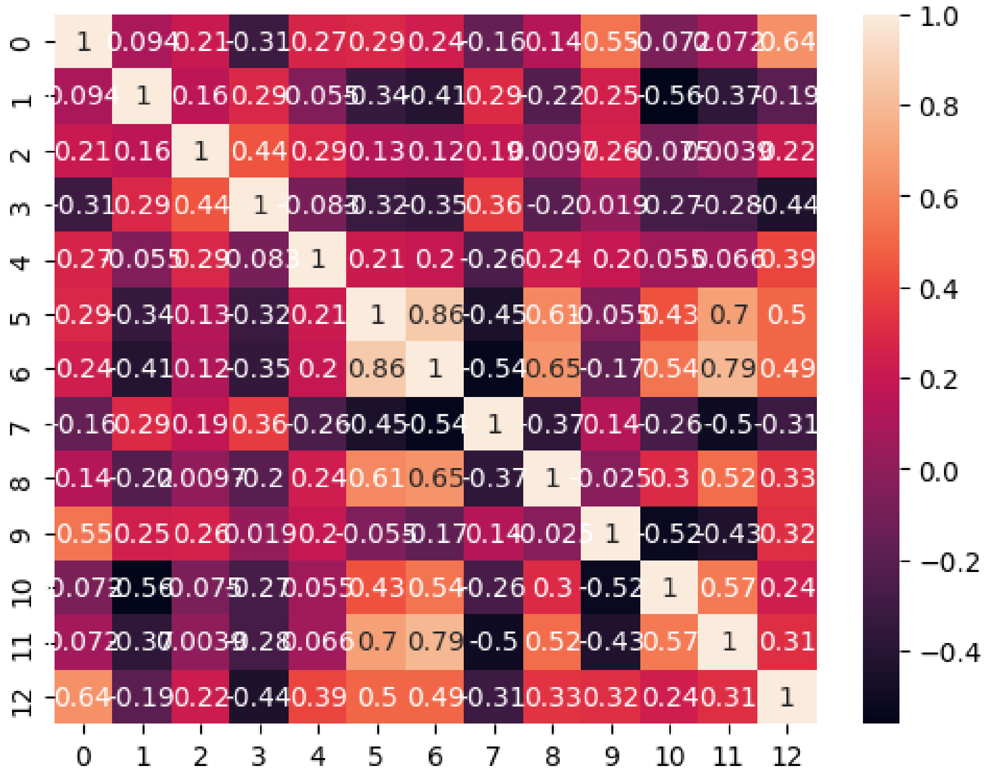
[5]:
```
cov_data = np.corrcoef(X.T)
cov_data
```

[5]:
```
array([[ 1.        ,  0.09439694,  0.2115446 , -0.31023514,  0.27079823,
         0.28910112,  0.23681493, -0.15592947,  0.13669791,  0.5463642 ,
        -0.0717472 ,  0.07234319,  0.64372004],
       [ 0.09439694,  1.        ,  0.16404547,  0.2885004 , -0.0545751 ,
        -0.335167  , -0.41100659,  0.29297713, -0.22074619,  0.24898534,
        -0.56129569, -0.36871043, -0.19201056],
       [ 0.2115446 ,  0.16404547,  1.        ,  0.44336719,  0.28658669,
         0.12897954,  0.11507728,  0.18623045,  0.00965194,  0.25888726,
        -0.07466689,  0.00391123,  0.22362626],
       [-0.31023514,  0.2885004 ,  0.44336719,  1.        , -0.08333309,
        -0.32111332, -0.35136986,  0.36192172, -0.19732684,  0.01873198,
        -0.27395522, -0.27676855, -0.44059693],
       [ 0.27079823, -0.0545751 ,  0.28658669, -0.08333309,  1.        ,
         0.21440123,  0.19578377, -0.25629405,  0.23644061,  0.19995001,
         0.0553982 ,  0.06600394,  0.39335085],
       [ 0.28910112, -0.335167  ,  0.12897954, -0.32111332,  0.21440123,
         1.        ,  0.8645635 , -0.4499353 ,  0.61241308, -0.05513642,
         0.43368134,  0.69994936,  0.49811488],
       [ 0.23681493, -0.41100659,  0.11507728, -0.35136986,  0.19578377,
```

```
  0.8645635 ,  1.        , -0.53789961,  0.65269177, -0.1723794 ,
  0.54347857,  0.7871939 ,  0.49419313],
[-0.15592947,  0.29297713,  0.18623045,  0.36192172, -0.25629405,
 -0.4499353 , -0.53789961,  1.        , -0.3658451 ,  0.13905701,
 -0.26263963, -0.5032696 , -0.31138519],
[ 0.13669791, -0.22074619,  0.00965194, -0.19732684,  0.23644061,
  0.61241308,  0.65269177, -0.3658451 ,  1.        , -0.02524993,
  0.29554425,  0.5190671 ,  0.3304167 ],
[ 0.5463642 ,  0.24898534,  0.25888726,  0.01873198,  0.19995001,
 -0.05513642, -0.1723794 ,  0.13905701, -0.02524993,  1.        ,
 -0.52181319, -0.42881494,  0.31610011],
[-0.0717472 , -0.56129569, -0.07466689, -0.27395522,  0.0553982 ,
  0.43368134,  0.54347857, -0.26263963,  0.29554425, -0.52181319,
  1.        ,  0.56546829,  0.23618345],
[ 0.07234319, -0.36871043,  0.00391123, -0.27676855,  0.06600394,
  0.69994936,  0.7871939 , -0.5032696 ,  0.5190671 , -0.42881494,
  0.56546829,  1.        ,  0.31276108],
[ 0.64372004, -0.19201056,  0.22362626, -0.44059693,  0.39335085,
  0.49811488,  0.49419313, -0.31138519,  0.3304167 ,  0.31610011,
  0.23618345,  0.31276108,  1.        ]])
```

[6]:
```python
import seaborn as sns

sns.heatmap(cov_data, annot=True)
```

[6]: <Axes: >



[7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=0)
```

[8]:
```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

[9]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)

X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

[10]:
```
explained_variance = pca.explained_variance_ratio_
```

[11]:
```
explained_variance
```

[11]:
```
array([0.36884109, 0.19318394])
```

[12]:
```
principal_Df = pd.DataFrame(data = X_train
             , columns = ['principal component 1', 'principal component 2'])
```

```
principal_Df
```

[12]:

```
     principal component 1  principal component 2
0                -2.178845               1.072185
1                -1.808192              -1.578223
2                 1.098295              -2.221243
3                -2.555847               1.662104
4                 1.856981              -0.241573
..                     …                      …
137              -0.501012              -2.684532
138               0.330454              -2.433962
139               0.010973              -1.995855
140               2.891767               0.771555
141              -2.448304               2.113603

[142 rows x 2 columns]
```

[13]:

```
from sklearn.pipeline import make_pipeline

# Entrenamiento modelo PCA con escalado de los datos
# ==============================================================================
pca_pipe = make_pipeline(sc, pca)
pca_pipe.fit(X)

# Se extrae el modelo entrenado del pipeline
modelo_pca = pca_pipe.named_steps['pca']
```

[14]:

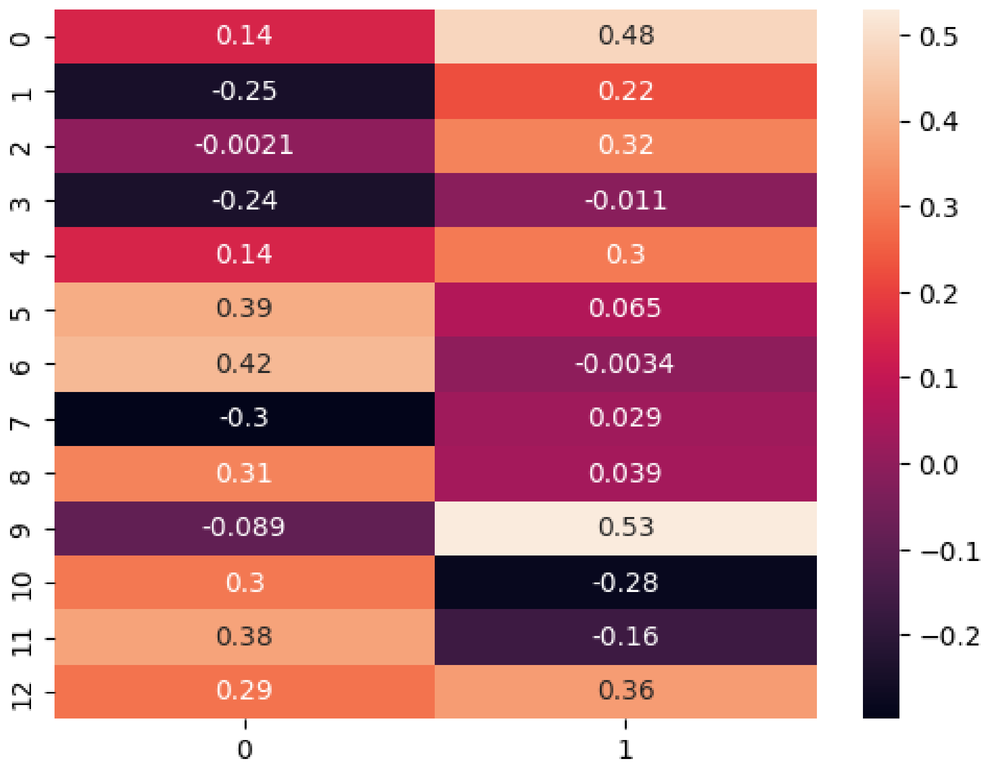
```
# Se combierte el array a dataframe para añadir nombres a los ejes.
pd.DataFrame(
    data    = modelo_pca.components_,
    columns = dataset.iloc[:, 0:13].columns,
    index   = ['PC1', 'PC2']
)
```

[14]:

```
      Alcohol  Malic_Acid       Ash  Ash_Alcanity  Magnesium  Total_Phenols  \
PC1  0.144329   -0.245188 -0.002051     -0.239320   0.141992       0.394661
PC2  0.483652    0.224931  0.316069     -0.010591   0.299634       0.065040

     Flavanoids  Nonflavanoid_Phenols  Proanthocyanins  Color_Intensity  \
PC1    0.422934             -0.298533         0.313429        -0.088617
PC2   -0.003360              0.028779         0.039302         0.529996

          Hue     OD280   Proline
PC1  0.296715  0.376167  0.286752
PC2 -0.279235 -0.164496  0.364903
```

[15]:
```python
componentes = modelo_pca.components_

sns.heatmap(componentes.T, annot=True)
```

[15]: <Axes: >



[16]:
```python
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
```

[16]: LogisticRegression(random_state=0)

[17]:
```python
y_pred = classifier.predict(X_test)
y_pred
```

[17]:
```
array([1, 3, 2, 1, 2, 1, 1, 3, 2, 2, 3, 3, 1, 2, 3, 2, 1, 1, 2, 1, 2, 1,
       1, 2, 2, 2, 2, 2, 2, 3, 1, 1, 2, 1, 1, 1])
```

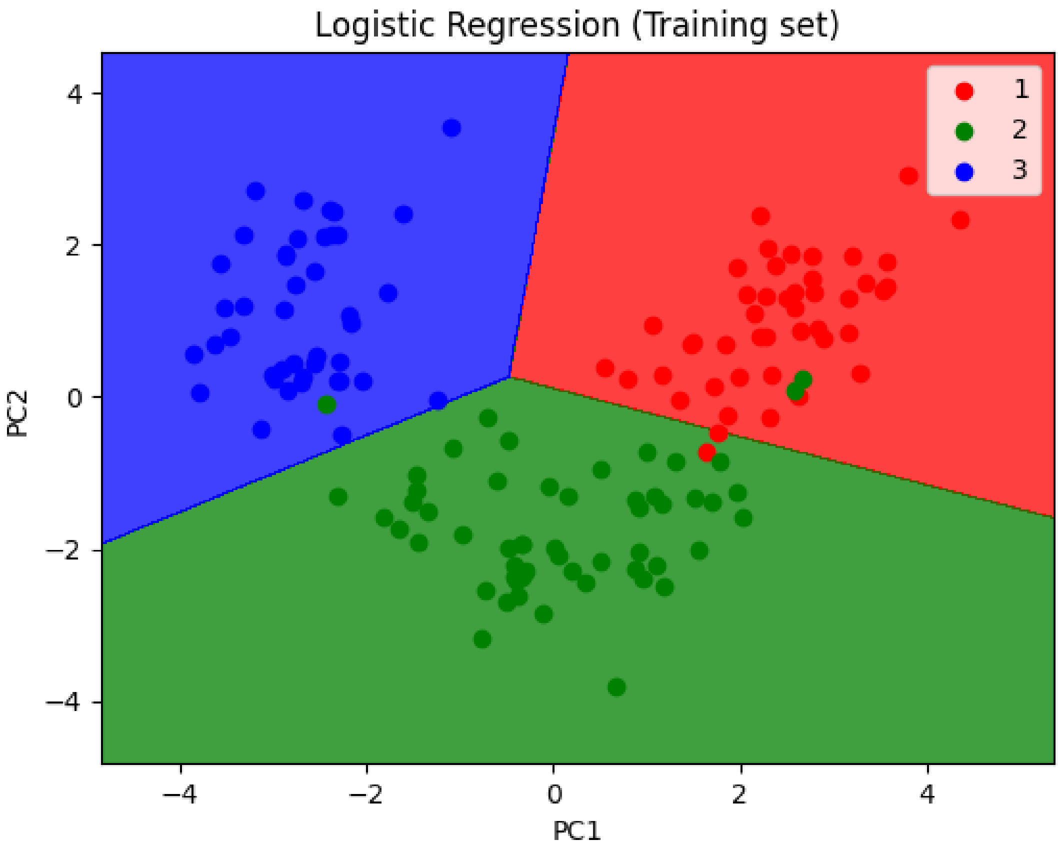
[18]:
```python
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
```

```
cm
```

[18]:
```
array([[14,  0,  0],
       [ 1, 15,  0],
       [ 0,  0,  6]])
```

[19]:
```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:,␣
 ↪0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:,␣
 ↪1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).
 ↪reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green', 'blue'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.show()
```

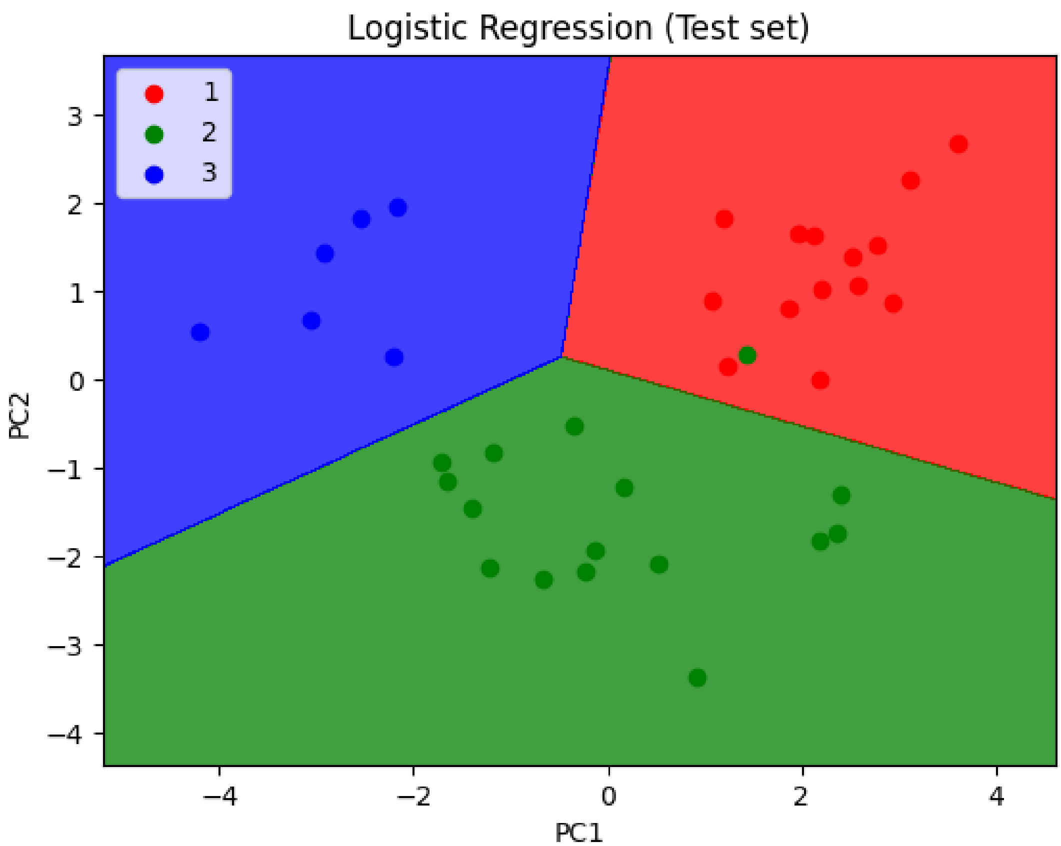
```
/tmp/ipykernel_11835/910810724.py:11: UserWarning: *c* argument looks like a
single numeric RGB or RGBA sequence, which should be avoided as value-mapping
will have precedence in case its length matches with *x* & *y*.  Please use the
*color* keyword-argument or provide a 2D array with a single row if you intend
to specify the same RGB or RGBA value for all points.
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
```



[20]:
```
# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:,␣
↪0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:,␣
↪1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).
↪reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green', 'blue'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
```

```
plt.show()
```

```
/tmp/ipykernel_11835/1176991918.py:11: UserWarning: *c* argument looks like a
single numeric RGB or RGBA sequence, which should be avoided as value-mapping
will have precedence in case its length matches with *x* & *y*.  Please use the
*color* keyword-argument or provide a 2D array with a single row if you intend
to specify the same RGB or RGBA value for all points.
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
```



Creado por:

Isabel Maniega