26th April Assignment

May 1, 2023

1 Assignment 79

[1]:
```
import pandas as pd
```

[3]:
```
# columns names
columns = ['Wine class','Alcohol','Malic_acid','Ash','Alcalinity of␣
↪ash','Magnesium','Total phenols','Flavanoids','Nonflavanoid␣
↪phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted␣
↪wines','Proline']
columns
```

[3]:
```
['Wine class',
 'Alcohol',
 'Malic_acid',
 'Ash',
 'Alcalinity of ash',
 'Magnesium',
 'Total phenols',
 'Flavanoids',
 'Nonflavanoid phenols',
 'Proanthocyanins',
 'Color intensity',
 'Hue',
 'OD280/OD315 of diluted wines',
 'Proline']
```

[4]:
```
# load the data
df = pd.read_csv('wine.data',names=columns)
df.shape
```

[4]:
```
(178, 14)
```

[5]:
```
df.head()
```

[5]:
```
   Wine class  Alcohol  Malic_acid   Ash  Alcalinity of ash  Magnesium  \
0           1    14.23        1.71  2.43               15.6        127
1           1    13.20        1.78  2.14               11.2        100
```

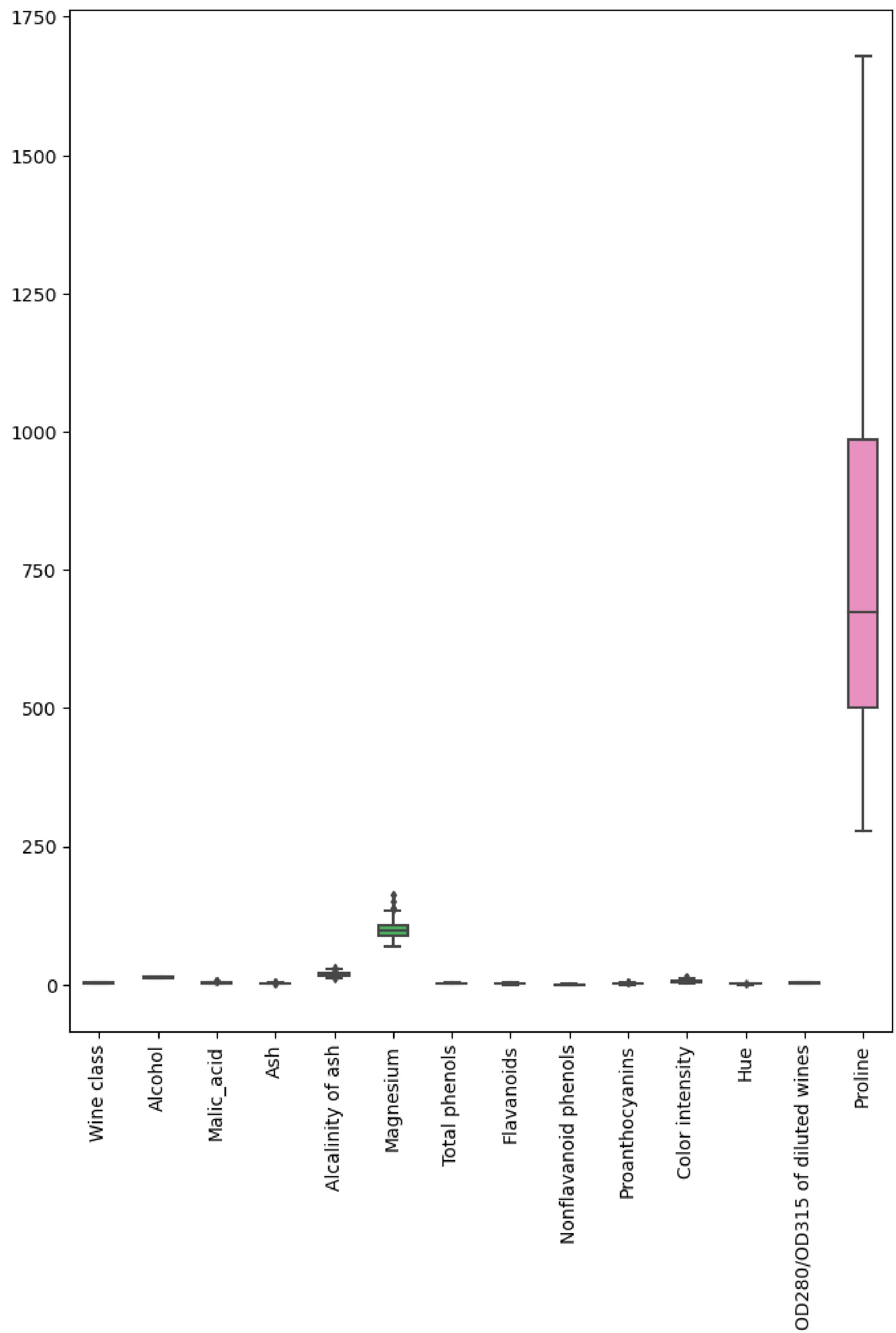
```
2            1     13.16         2.36   2.67                18.6         101
3            1     14.37         1.95   2.50                16.8         113
4            1     13.24         2.59   2.87                21.0         118

   Total phenols  Flavanoids  Nonflavanoid phenols  Proanthocyanins  \
0           2.80        3.06                  0.28             2.29
1           2.65        2.76                  0.26             1.28
2           2.80        3.24                  0.30             2.81
3           3.85        3.49                  0.24             2.18
4           2.80        2.69                  0.39             1.82

   Color intensity   Hue  OD280/OD315 of diluted wines  Proline
0             5.64  1.04                          3.92     1065
1             4.38  1.05                          3.40     1050
2             5.68  1.03                          3.17     1185
3             7.80  0.86                          3.45     1480
4             4.32  1.04                          2.93      735
```

[6]:
```python
df.isnull().sum()
```

[6]:
```
Wine class                      0
Alcohol                         0
Malic_acid                      0
Ash                             0
Alcalinity of ash               0
Magnesium                       0
Total phenols                   0
Flavanoids                      0
Nonflavanoid phenols            0
Proanthocyanins                 0
Color intensity                 0
Hue                             0
OD280/OD315 of diluted wines    0
Proline                         0
dtype: int64
```

[7]:
```python
# check for outliers in dataset
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

[8]:
```python
plt.figure(figsize=(8,10))
sns.boxplot(data=df,fliersize=3,width=0.5)
plt.xticks(rotation=90)
plt.show()
```


1750
1500
1250
1000
750
500
250
0
Wine class
Alcohol
Malic_acid
Ash
Alcalinity of ash
Magnesium
Total phenols
Flavanoids
Nonflavanoid phenols
Proanthocyanins
Color intensity
Hue
OD280/OD315 of diluted wines
Proline

[9]:
```
# segregate the data into independent and dependent
X = df.drop(columns='Wine class',axis=1)
y = df['Wine class']
```

[10]:
```
# standardization for removing outliers
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

[11]:
```
X = scaler.fit_transform(X)
```

[14]:
```
X
```

[14]:
```
array([[ 1.51861254, -0.5622498 ,  0.23205254, …,  0.36217728,
         1.84791957,  1.01300893],
       [ 0.24628963, -0.49941338, -0.82799632, …,  0.40605066,
         1.1134493 ,  0.96524152],
       [ 0.19687903,  0.02123125,  1.10933436, …,  0.31830389,
         0.78858745,  1.39514818],
       …,
       [ 0.33275817,  1.74474449, -0.38935541, …, -1.61212515,
        -1.48544548,  0.28057537],
       [ 0.20923168,  0.22769377,  0.01273209, …, -1.56825176,
        -1.40069891,  0.29649784],
       [ 1.39508604,  1.58316512,  1.36520822, …, -1.52437837,
        -1.42894777, -0.59516041]])
```

[20]:
```
# pca
from sklearn.decomposition import PCA
pca = PCA()
```

[21]:
```
X = pca.fit_transform(X)
```

[22]:
```
X
```

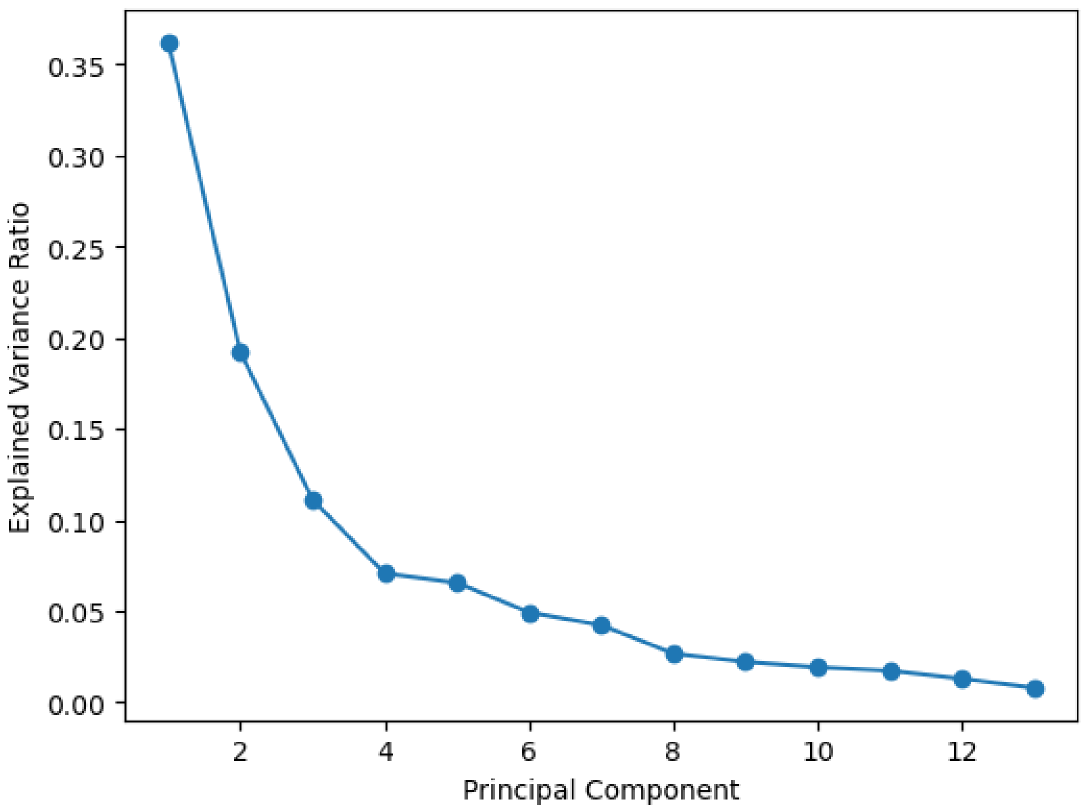
[22]:
```
array([[ 3.31675081e+00, -1.44346263e+00, -1.65739045e-01, …,
        -4.51563395e-01,  5.40810414e-01, -6.62386309e-02],
       [ 2.20946492e+00,  3.33392887e-01, -2.02645737e+00, …,
        -1.42657306e-01,  3.88237741e-01,  3.63650247e-03],
       [ 2.51674015e+00, -1.03115130e+00,  9.82818670e-01, …,
        -2.86672847e-01,  5.83573183e-04,  2.17165104e-02],
       …,
       [-2.67783946e+00, -2.76089913e+00, -9.40941877e-01, …,
         5.12492025e-01,  6.98766451e-01,  7.20776948e-02],
       [-2.38701709e+00, -2.29734668e+00, -5.50696197e-01, …,
         2.99821968e-01,  3.39820654e-01, -2.18657605e-02],
       [-3.20875816e+00, -2.76891957e+00,  1.01391366e+00, …,
```

```
-2.29964331e-01, -1.88787963e-01, -3.23964720e-01]])
```

[23]:
```
pca.explained_variance_ratio_
```

[23]:
```
array([0.36198848, 0.1920749 , 0.11123631, 0.0706903 , 0.06563294,
       0.04935823, 0.04238679, 0.02680749, 0.02222153, 0.01930019,
       0.01736836, 0.01298233, 0.00795215])
```

[24]:
```
# To determine the optimal number of principal components to retain, we will␣
↪plot the explained variance ratio.

import matplotlib.pyplot as plt
plt.plot(range(1, len(pca.explained_variance_ratio_)+1), pca.
↪explained_variance_ratio_, marker='o')
plt.xlabel('Principal Component')
plt.ylabel('Explained Variance Ratio')
plt.show()
```



[26]:
```
# We can see majority variance captured at 2
pca = PCA(n_components=2)
```

[27]:
```python
X = pca.fit_transform(X)
```

[28]:
```python
X
```

[28]:
```
array([[ 3.31675081, -1.44346263],
       [ 2.20946492,  0.33339289],
       [ 2.51674015, -1.0311513 ],
       [ 3.75706561, -2.75637191],
       [ 1.00890849, -0.86983082],
       [ 3.05025392, -2.12240111],
       [ 2.44908967, -1.17485013],
       [ 2.05943687, -1.60896307],
       [ 2.5108743 , -0.91807096],
       [ 2.75362819, -0.78943767],
       [ 3.47973668, -1.30233324],
       [ 1.7547529 , -0.61197723],
       [ 2.11346234, -0.67570634],
       [ 3.45815682, -1.13062988],
       [ 4.31278391, -2.09597558],
       [ 2.3051882 , -1.66255173],
       [ 2.17195527, -2.32730534],
       [ 1.89897118, -1.63136888],
       [ 3.54198508, -2.51834367],
       [ 2.0845222 , -1.06113799],
       [ 3.12440254, -0.78689711],
       [ 1.08657007, -0.24174355],
       [ 2.53522408,  0.09184062],
       [ 1.64498834,  0.51627893],
       [ 1.76157587,  0.31714893],
       [ 0.9900791 , -0.94066734],
       [ 1.77527763, -0.68617513],
       [ 1.23542396,  0.08980704],
       [ 2.18840633, -0.68956962],
       [ 2.25610898, -0.19146194],
       [ 2.50022003, -1.24083383],
       [ 2.67741105, -1.47187365],
       [ 1.62857912, -0.05270445],
       [ 1.90269086, -1.63306043],
       [ 1.41038853, -0.69793432],
       [ 1.90382623, -0.17671095],
       [ 1.38486223, -0.65863985],
       [ 1.12220741, -0.11410976],
       [ 1.5021945 ,  0.76943201],
       [ 2.52980109, -1.80300198],
       [ 2.58809543, -0.7796163 ],
       [ 0.66848199, -0.16996094],
       [ 3.07080699, -1.15591896],
```

```
[ 0.46220914, -0.33074213],
[ 2.10135193,  0.07100892],
[ 1.13616618, -1.77710739],
[ 2.72660096, -1.19133469],
[ 2.82133927, -0.6462586 ],
[ 2.00985085, -1.24702946],
[ 2.7074913 , -1.75196741],
[ 3.21491747, -0.16699199],
[ 2.85895983, -0.7452788 ],
[ 3.50560436, -1.61273386],
[ 2.22479138, -1.875168  ],
[ 2.14698782, -1.01675154],
[ 2.46932948, -1.32900831],
[ 2.74151791, -1.43654878],
[ 2.17374092, -1.21219984],
[ 3.13938015, -1.73157912],
[-0.92858197,  3.07348616],
[-1.54248014,  1.38144351],
[-1.83624976,  0.82998412],
[ 0.03060683,  1.26278614],
[ 2.05026161,  1.9250326 ],
[-0.60968083,  1.90805881],
[ 0.90022784,  0.76391147],
[ 2.24850719,  1.88459248],
[ 0.18338403,  2.42714611],
[-0.81280503,  0.22051399],
[ 1.9756205 ,  1.40328323],
[-1.57221622,  0.88498314],
[ 1.65768181,  0.9567122 ],
[-0.72537239,  1.0636454 ],
[ 2.56222717, -0.26019855],
[ 1.83256757,  1.2878782 ],
[-0.8679929 ,  2.44410119],
[ 0.3700144 ,  2.15390698],
[-1.45737704,  1.38335177],
[ 1.26293085,  0.77084953],
[ 0.37615037,  1.0270434 ],
[ 0.7620639 ,  3.37505381],
[ 1.03457797,  1.45070974],
[-0.49487676,  2.38124353],
[-2.53897708,  0.08744336],
[ 0.83532015,  1.47367055],
[ 0.78790461,  2.02662652],
[-0.80683216,  2.23383039],
[-0.55804262,  2.37298543],
[-1.11511104,  1.80224719],
[-0.55572283,  2.65754004],
```

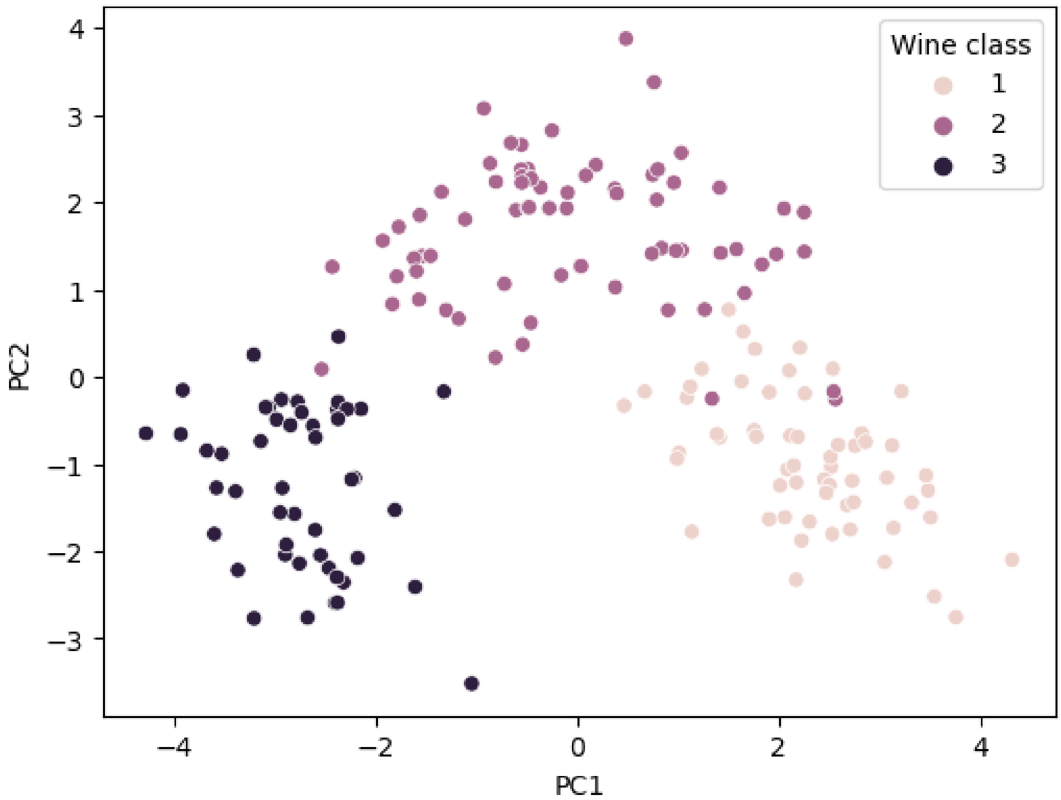
```
[-1.34928528,  2.11800147],
[-1.56448261,  1.85221452],
[-1.93255561,  1.55949546],
[ 0.74666594,  2.31293171],
[ 0.95745536,  2.22352843],
[ 2.54386518, -0.16927402],
[-0.54395259,  0.36892655],
[ 1.03104975,  2.56556935],
[ 2.25190942,  1.43274138],
[ 1.41021602,  2.16619177],
[ 0.79771979,  2.3769488 ],
[-0.54953173,  2.29312864],
[-0.16117374,  1.16448332],
[-0.65979494,  2.67996119],
[ 0.39235441,  2.09873171],
[-1.77249908,  1.71728847],
[-0.36626736,  2.1693533 ],
[-1.62067257,  1.35558339],
[ 0.08253578,  2.30623459],
[ 1.57827507,  1.46203429],
[ 1.42056925,  1.41820664],
[-0.27870275,  1.93056809],
[-1.30314497,  0.76317231],
[-0.45707187,  2.26941561],
[-0.49418585,  1.93904505],
[ 0.48207441,  3.87178385],
[-0.25288888,  2.82149237],
[-0.10722764,  1.92892204],
[-2.4330126 ,  1.25714104],
[-0.55108954,  2.22216155],
[ 0.73962193,  1.40895667],
[ 1.33632173, -0.25333693],
[-1.177087  ,  0.66396684],
[-0.46233501,  0.61828818],
[ 0.97847408,  1.4455705 ],
[-0.09680973,  2.10999799],
[ 0.03848715,  1.26676211],
[-1.5971585 ,  1.20814357],
[-0.47956492,  1.93884066],
[-1.79283347,  1.1502881 ],
[-1.32710166, -0.17038923],
[-2.38450083, -0.37458261],
[-2.9369401 , -0.26386183],
[-2.14681113, -0.36825495],
[-2.36986949,  0.45963481],
[-3.06384157, -0.35341284],
[-3.91575378, -0.15458252],
```

```
       [-3.93646339, -0.65968723],
       [-3.09427612, -0.34884276],
       [-2.37447163, -0.29198035],
       [-2.77881295, -0.28680487],
       [-2.28656128, -0.37250784],
       [-2.98563349, -0.48921791],
       [-2.3751947 , -0.48233372],
       [-2.20986553, -1.1600525 ],
       [-2.625621  , -0.56316076],
       [-4.28063878, -0.64967096],
       [-3.58264137, -1.27270275],
       [-2.80706372, -1.57053379],
       [-2.89965933, -2.04105701],
       [-2.32073698, -2.35636608],
       [-2.54983095, -2.04528309],
       [-1.81254128, -1.52764595],
       [-2.76014464, -2.13893235],
       [-2.7371505 , -0.40988627],
       [-3.60486887, -1.80238422],
       [-2.889826  , -1.92521861],
       [-3.39215608, -1.31187639],
       [-1.0481819 , -3.51508969],
       [-1.60991228, -2.40663816],
       [-3.14313097, -0.73816104],
       [-2.2401569 , -1.17546529],
       [-2.84767378, -0.55604397],
       [-2.59749706, -0.69796554],
       [-2.94929937, -1.55530896],
       [-3.53003227, -0.8825268 ],
       [-2.40611054, -2.59235618],
       [-2.92908473, -1.27444695],
       [-2.18141278, -2.07753731],
       [-2.38092779, -2.58866743],
       [-3.21161722,  0.2512491 ],
       [-3.67791872, -0.84774784],
       [-2.4655558 , -2.1937983 ],
       [-3.37052415, -2.21628914],
       [-2.60195585, -1.75722935],
       [-2.67783946, -2.76089913],
       [-2.38701709, -2.29734668],
       [-3.20875816, -2.76891957]])
```

[29]:

```
import seaborn as sns

sns.scatterplot(x=X[:,0], y=X[:,1], hue=y)
plt.xlabel('PC1')
plt.ylabel('PC2')
```

```
plt.show()
```



1.0.1 Model training

[30]:
```
from sklearn.cluster import KMeans
```

[31]:
```
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

[31]: KMeans(n_clusters=3)

[32]:
```
# Display the performance metrics for the clustering algorithm
from sklearn.metrics import silhouette_score
metrics = pd.DataFrame({'Metric': ['Inertia', 'Silhouette Score'],
                        'Score': [kmeans.inertia_, silhouette_score(X, kmeans.
 ↪labels_)]})
```

```python
print(metrics)
```

```
             Metric       Score
0           Inertia  259.509381
1  Silhouette Score    0.561051
```

1.0.2 Report:

The wine dataset was downloaded from the UCI Machine Learning Repository and loaded into a Pandas dataframe. The dataset contains 178 instances and 13 attributes, including the class attribute.

Data preprocessing was performed to scale the data using the StandardScaler from the scikit-learn library. Principal Component Analysis (PCA) was then performed on the preprocessed dataset using the PCA from the scikit-learn library.

A plot of the explained variance ratio showed that the first two principal components explain the majority of the variance in the data. Therefore, the first two principal components were retained for further analysis.