Wines

January 4, 2023

```
[237]: import random
       import pandas as pd
       import seaborn as sns
       import matplotlib.pyplot as plt
       import numpy as np
       from sklearn.preprocessing import StandardScaler
       from sklearn.preprocessing import MinMaxScaler
       from sklearn.decomposition import PCA
       from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
       import matplotlib.pyplot as plt
       import plotly.express as px
       import plotly.graph_objects as go
```

1 Wine dataset

Dataset column names

0. Wine class
1. Alcohol: The alcohol content of the wine, measured in % vol.
2. Malic acid: The amount of malic acid in the wine, measured in g/l.
3. Ash: The ash content of the wine, measured in g/l.
4. Alcalinity of ash: The alkalinity of the ash in the wine, measured in g/l.
5. Magnesium: The amount of magnesium in the wine, measured in mg/l.
6. Total phenols: The total phenol content of the wine, measured in mg/l.
7. Flavanoids: The flavanoid content of the wine, measured in mg/l.
8. Nonflavanoid phenols: The nonflavanoid phenol content of the wine, measured in mg/l.
9. Proanthocyanins: The proanthocyanin content of the wine, measured in mg/l.
10. Color intensity: The color intensity of the wine, measured in absorbance units.
11. Hue: The hue of the wine, measured in absorbance units.
12. OD280/OD315 of diluted wines: The OD280/OD315 ratio of the diluted wine, which is a measure of the wine's color intensity.
13. Proline: The proline content of the wine, measured in mg/l.

Loading dataset

```
[5]: url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data'

     df = pd.read_csv(url, header=None)
```

```
df.columns = ['Class', 'Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash',␣
 ↪'Magnesium',
              'Total phenols', 'Flavanoids', 'Nonflavanoid phenols',␣
 ↪'Proanthocyanins',
              'Color intensity', 'Hue', 'OD280/OD315 of diluted wines',␣
 ↪'Proline']
```

Dataset preview and descriptive statistics

[6]:
```
df.head()
```

[6]:
```
   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
```

[7]:
```
df.describe()
```

[7]:
```
            Class     Alcohol  Malic acid         Ash  Alcalinity of ash  \
count  178.000000  178.000000  178.000000  178.000000         178.000000
mean     1.938202   13.000618    2.336348    2.366517          19.494944
std      0.775035    0.811827    1.117146    0.274344           3.339564
min      1.000000   11.030000    0.740000    1.360000          10.600000
25%      1.000000   12.362500    1.602500    2.210000          17.200000
50%      2.000000   13.050000    1.865000    2.360000          19.500000
75%      3.000000   13.677500    3.082500    2.557500          21.500000
max      3.000000   14.830000    5.800000    3.230000          30.000000

        Magnesium  Total phenols  Flavanoids  Nonflavanoid phenols  \
count  178.000000     178.000000  178.000000            178.000000
mean    99.741573       2.295112    2.029270              0.361854
```

```
std     14.282484       0.625851    0.998859              0.124453
min     70.000000       0.980000    0.340000              0.130000
25%     88.000000       1.742500    1.205000              0.270000
50%     98.000000       2.355000    2.135000              0.340000
75%    107.000000       2.800000    2.875000              0.437500
max    162.000000       3.880000    5.080000              0.660000

       Proanthocyanins  Color intensity         Hue  \
count       178.000000       178.000000  178.000000
mean          1.590899         5.058090    0.957449
std           0.572359         2.318286    0.228572
min           0.410000         1.280000    0.480000
25%           1.250000         3.220000    0.782500
50%           1.555000         4.690000    0.965000
75%           1.950000         6.200000    1.120000
max           3.580000        13.000000    1.710000

       OD280/OD315 of diluted wines      Proline
count                    178.000000   178.000000
mean                       2.611685   746.893258
std                        0.709990   314.907474
min                        1.270000   278.000000
25%                        1.937500   500.500000
50%                        2.780000   673.500000
75%                        3.170000   985.000000
max                        4.000000  1680.000000
```

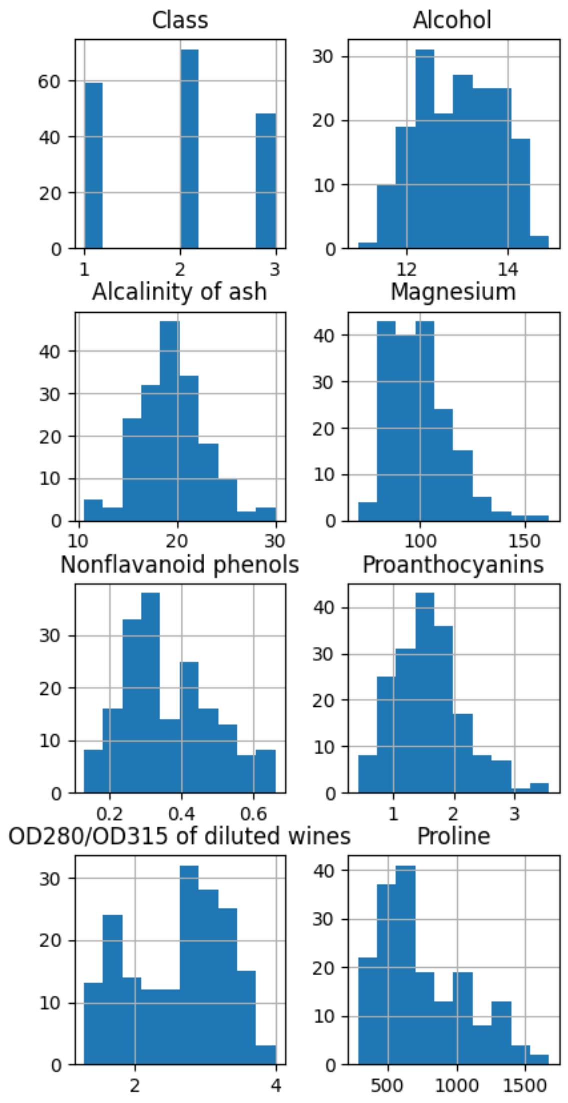
Check for NaN values

[9]:
```
df.isna().sum()
```

[9]:
```
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
```

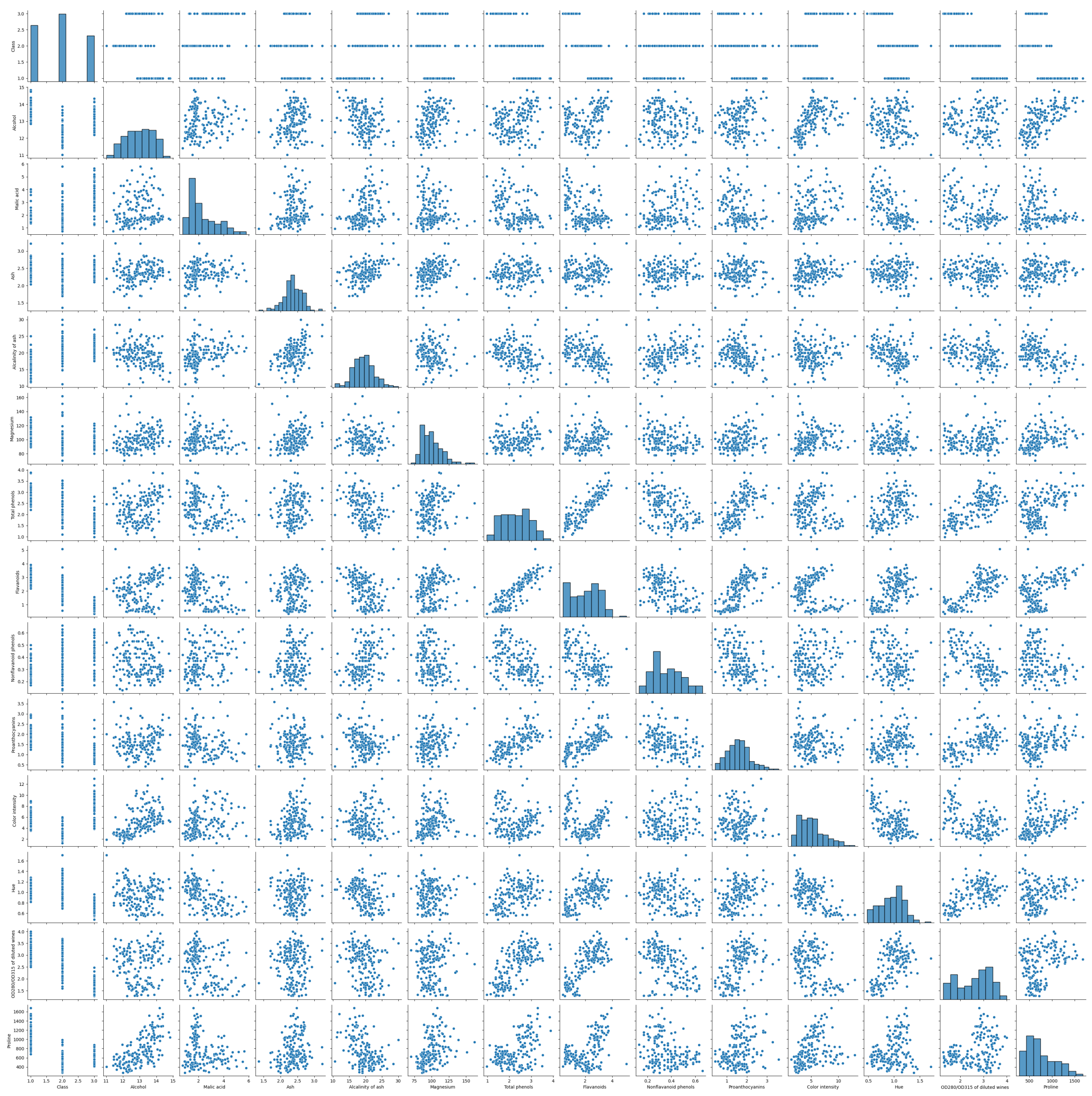
Exploring data distributions

[11]:
```
df.hist(figsize=(10,10))
plt.show()
```



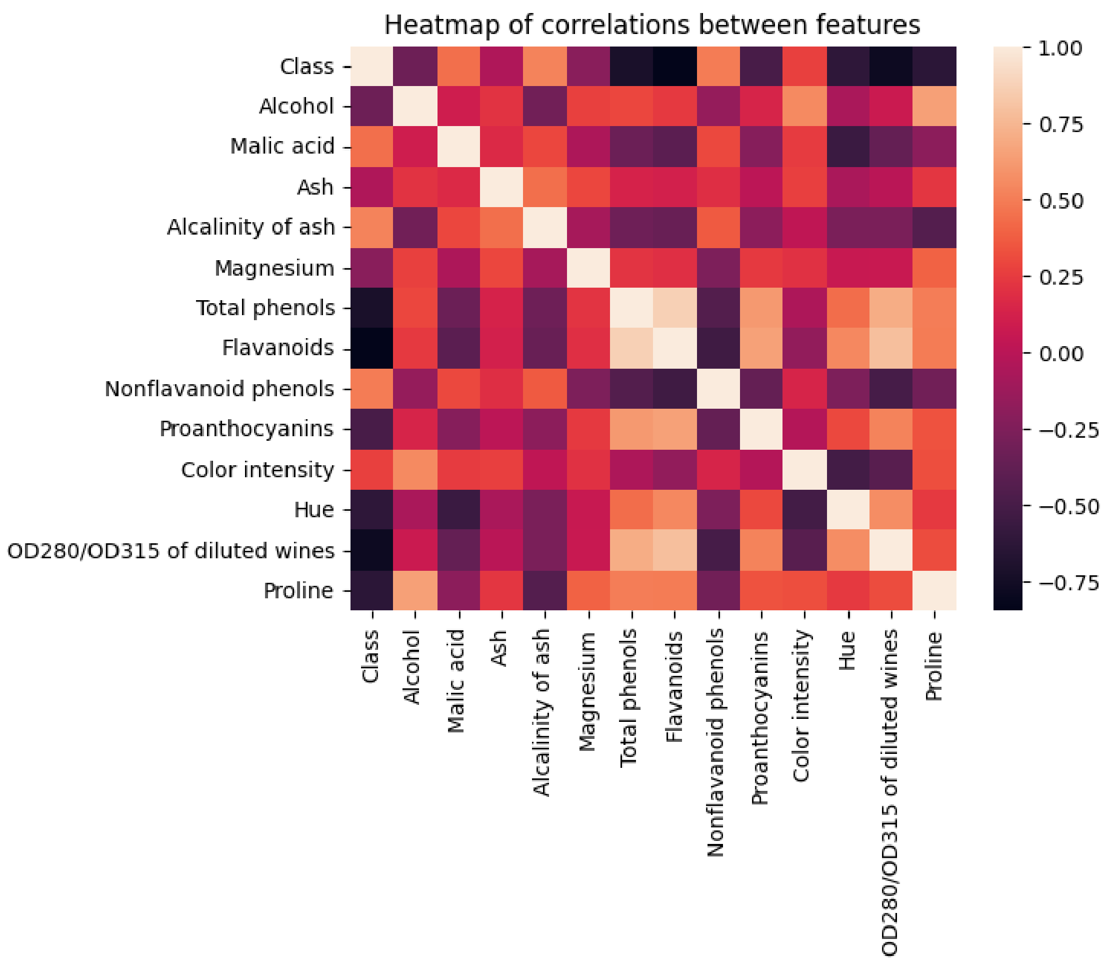
Visualizing the relationships between pairs of different features

[11]:
```
sns.pairplot(df)
plt.show()
```



Visualizing the correlations

[251]:
```
correlations = df.corr()
sns.heatmap(correlations, annot=False)

plt.title('Heatmap of correlations between features')
plt.show()
```



1.0.1 Perform PCA to reduce dimensionality

Scalling data Separate the features from the class labels and scale data for PCA analysis

[239]:
```
X = df.iloc[:, 1:]
y = df.iloc[:, 0]

scaler = StandardScaler()

scaler.fit(X)

X_scaled = scaler.transform(X)
```

[240]:
```
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)
```

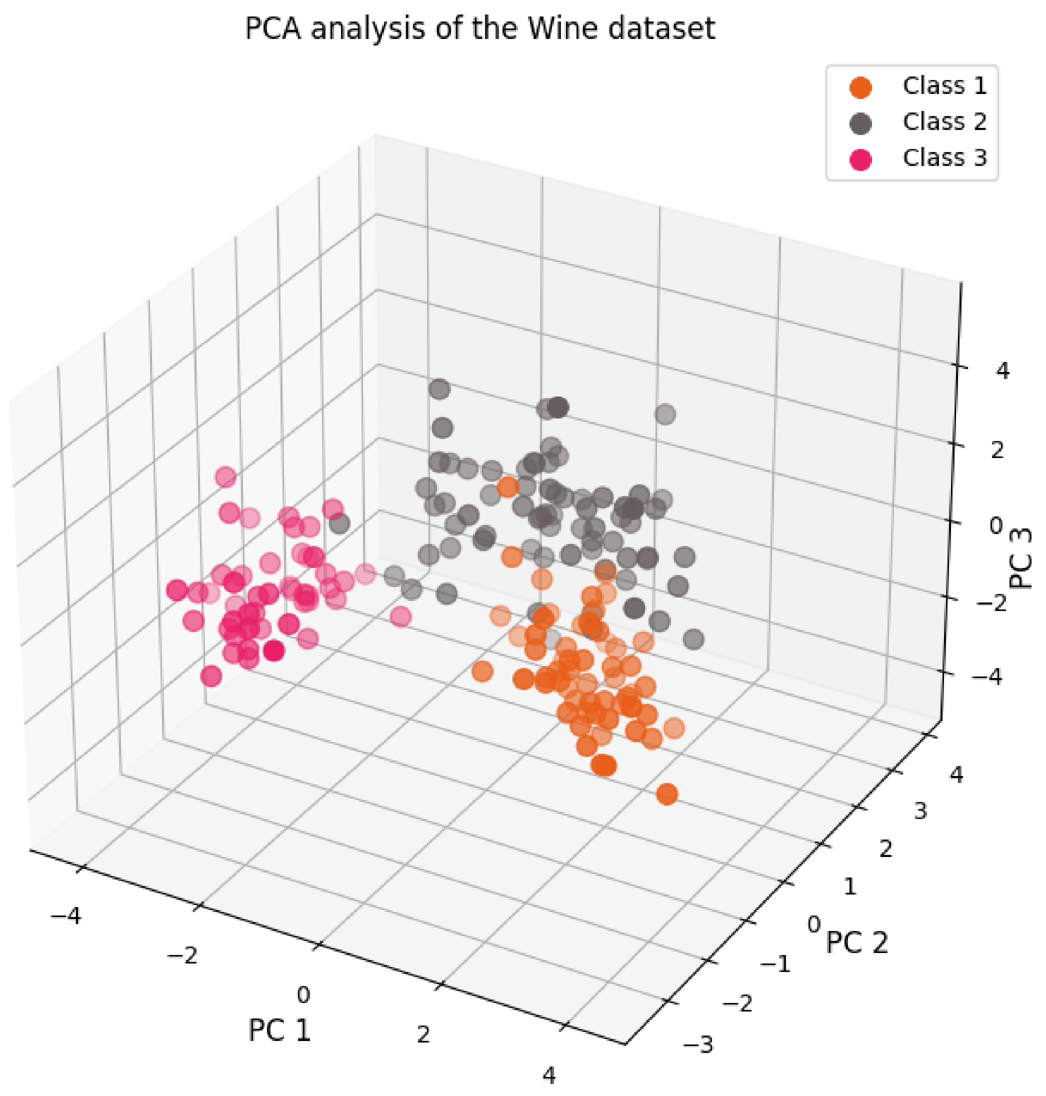
Get the explained variance ratio

```
[191]: explained_variance_ratio = pca.explained_variance_ratio_

       print(explained_variance_ratio)
```

```
[0.36198848 0.1920749  0.11123631]
```

```
[112]: cc = ColorConverter()
       colors = [cc.to_rgba(c) for c in np.random.rand(len(np.unique(y)), 3)]
```

```
[192]: fig = plt.figure(figsize=(20, 8))
       ax = fig.add_subplot(111, projection='3d')
       for i, label in enumerate(np.unique(y)):
           ax.scatter(X_pca[y==label, 0], X_pca[y==label, 1], X_pca[y==label, 2],␣
        ↪color=colors[i], s=70)

       ax.set_xlabel('PC 1', fontsize=12, labelpad=5)
       ax.set_ylabel('PC 2', fontsize=12, labelpad=5)
       ax.text2D(1.0, 0.5, 'PC 3', transform=ax.transAxes, fontsize=12, rotation=90)

       labels = ['Class 1', 'Class 2', 'Class 3']
       ax.legend(labels)
       ax.set_title('PCA analysis of the Wine dataset')

       plt.show()
```

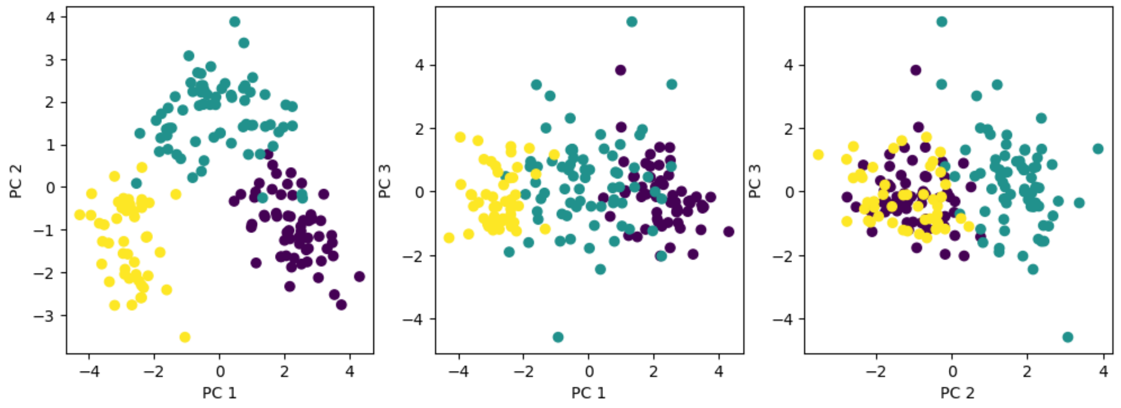


1.0.2 Interactive 3D Scatter Plot

[250]:
```
fig = px.scatter_3d(x=X_pca[:,0], y=X_pca[:,1], z=X_pca[:,2],␣
  ↪title='Interactive 3D Scatter Plot', labels={'x':'PC 1', 'y':'PC 2', 'z':'PC␣
  ↪3'}, color=y)

fig.show()
```

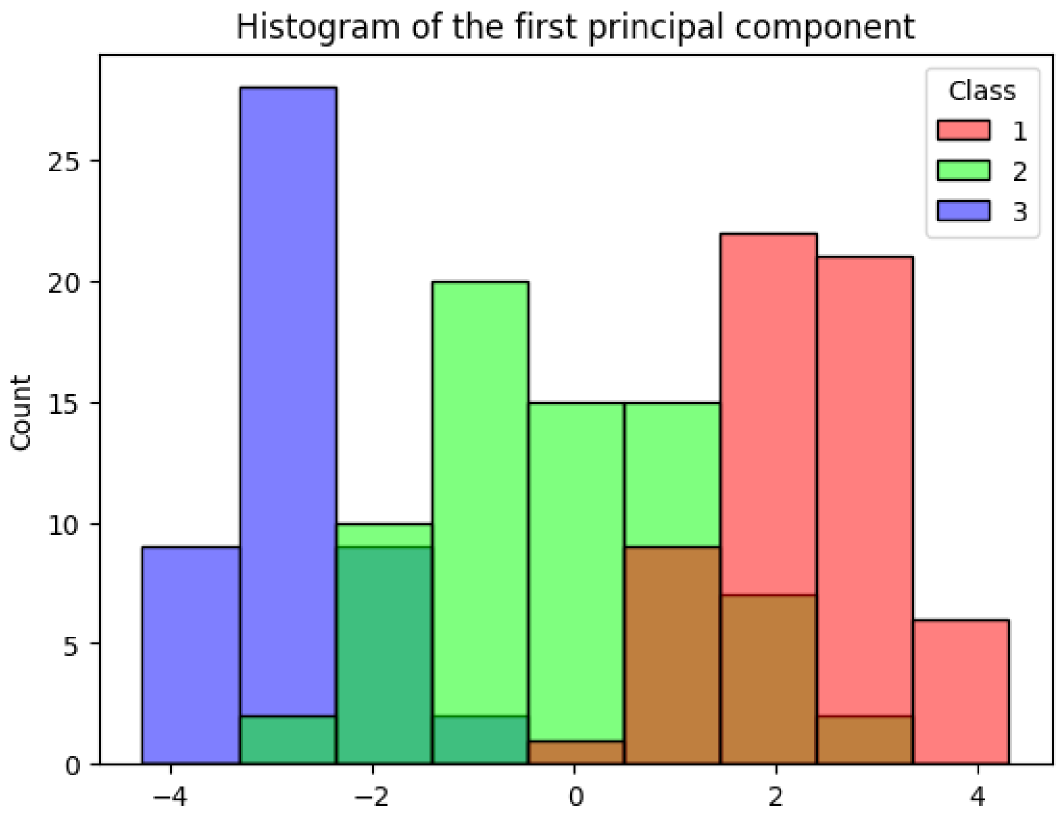
[205]:
```
fig, ax = plt.subplots(1, 3, figsize=(12,4))

# Scatter plot of PC 1 vs PC 2
ax[0].scatter(X_pca[:,0], X_pca[:,1], c=y)
```

```
ax[0].set_xlabel('PC 1')
ax[0].set_ylabel('PC 2')

# Scatter plot of PC 1 vs PC 3
ax[1].scatter(X_pca[:,0], X_pca[:,2], c=y)
ax[1].set_xlabel('PC 1')
ax[1].set_ylabel('PC 3')

# Scatter plot of PC 2 vs PC 3
ax[2].scatter(X_pca[:,1], X_pca[:,2], c=y)
ax[2].set_xlabel('PC 2')
ax[2].set_ylabel('PC 3')

# Show the plot
plt.show()
```

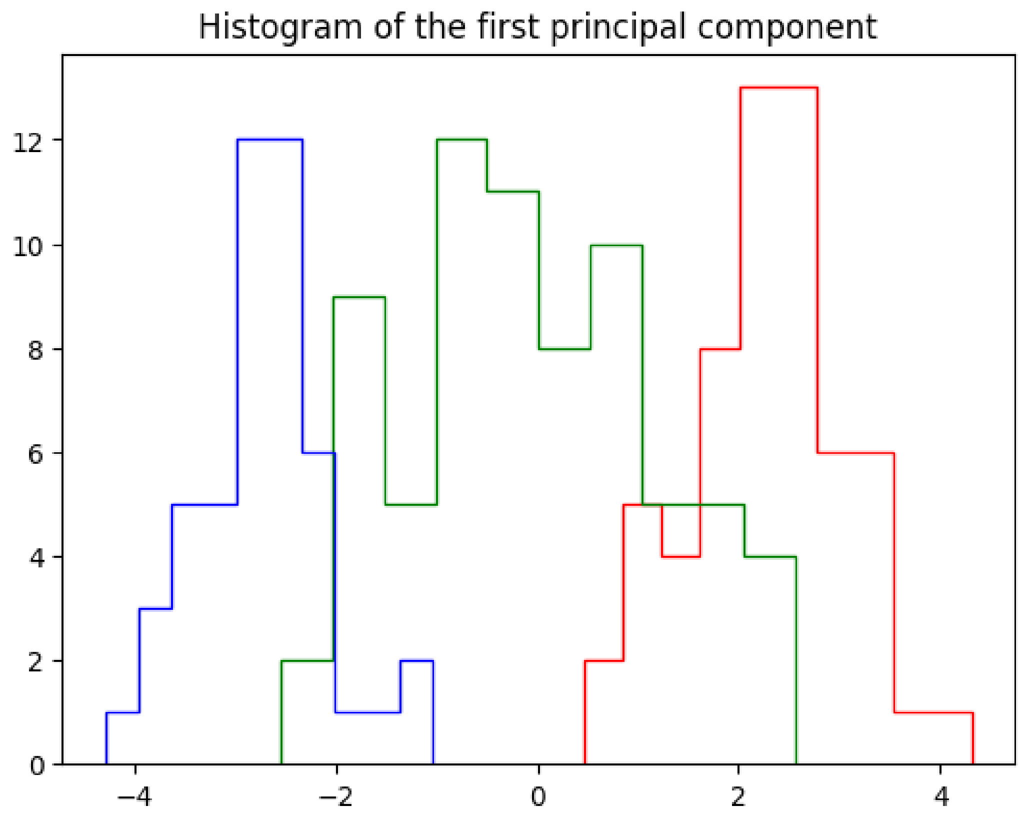


Create a histogram of the first principal component and histograms for each class

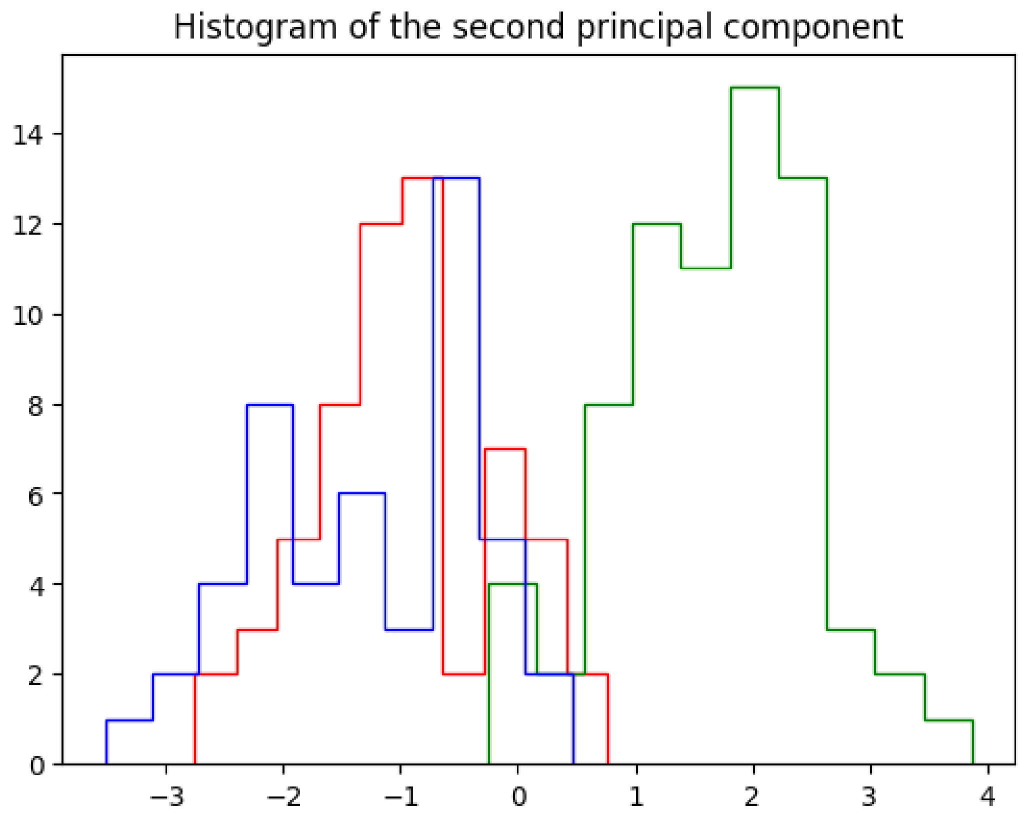
[220]:
```
sns.histplot(x=X_pca[:, 0], hue=y, palette=[(1, 0, 0), (0, 1, 0), (0, 0, 1)])

plt.title('Histogram of the first principal component')

plt.show()
```



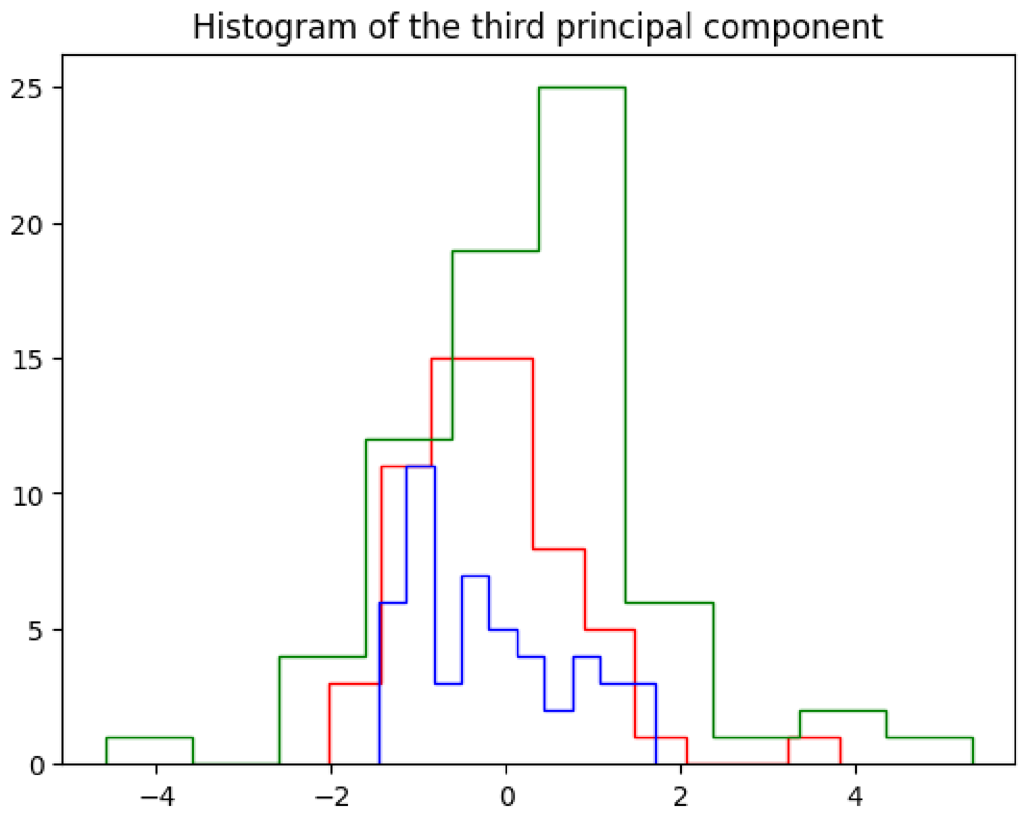
[212]:
```
# Select the samples for each class
class_1 = X_pca[y == 1, 0]
class_2 = X_pca[y == 2, 0]
class_3 = X_pca[y == 3, 0]

# Create a histogram for each class
plt.hist(class_1, histtype='step', color='red')
plt.hist(class_2, histtype='step', color='green')
plt.hist(class_3, histtype='step', color='blue')

# Show the plot
plt.title('Histogram of the first principal component')
plt.show()
```



[213]:
```python
# Select the samples for each class
class_1 = X_pca[y == 1, 1]
class_2 = X_pca[y == 2, 1]
class_3 = X_pca[y == 3, 1]

# Create a histogram for each class
plt.hist(class_1, histtype='step', color='red')
plt.hist(class_2, histtype='step', color='green')
plt.hist(class_3, histtype='step', color='blue')

# Show the plot
plt.title('Histogram of the second principal component')
plt.show()
```

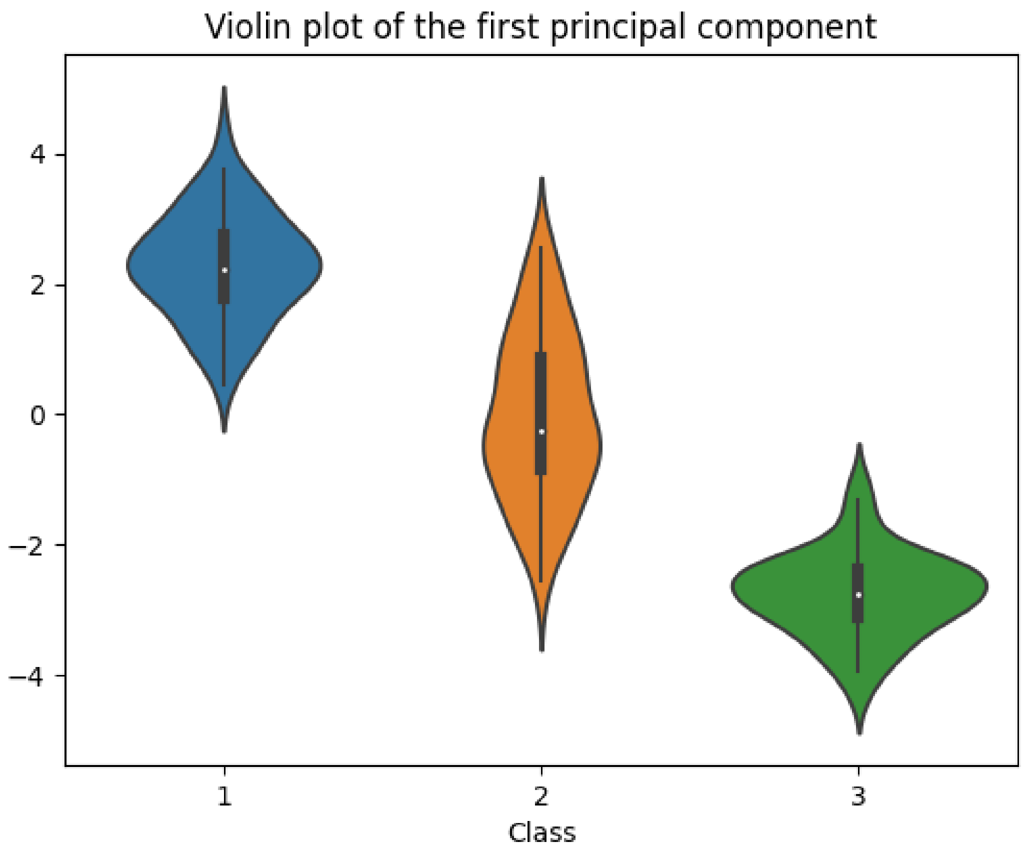


[214]:
```
# Select the samples for each class
class_1 = X_pca[y == 1, 2]
class_2 = X_pca[y == 2, 2]
class_3 = X_pca[y == 3, 2]

# Create a histogram for each class
plt.hist(class_1, histtype='step', color='red')
plt.hist(class_2, histtype='step', color='green')
plt.hist(class_3, histtype='step', color='blue')

# Show the plot
plt.title('Histogram of the third principal component')
plt.show()
```



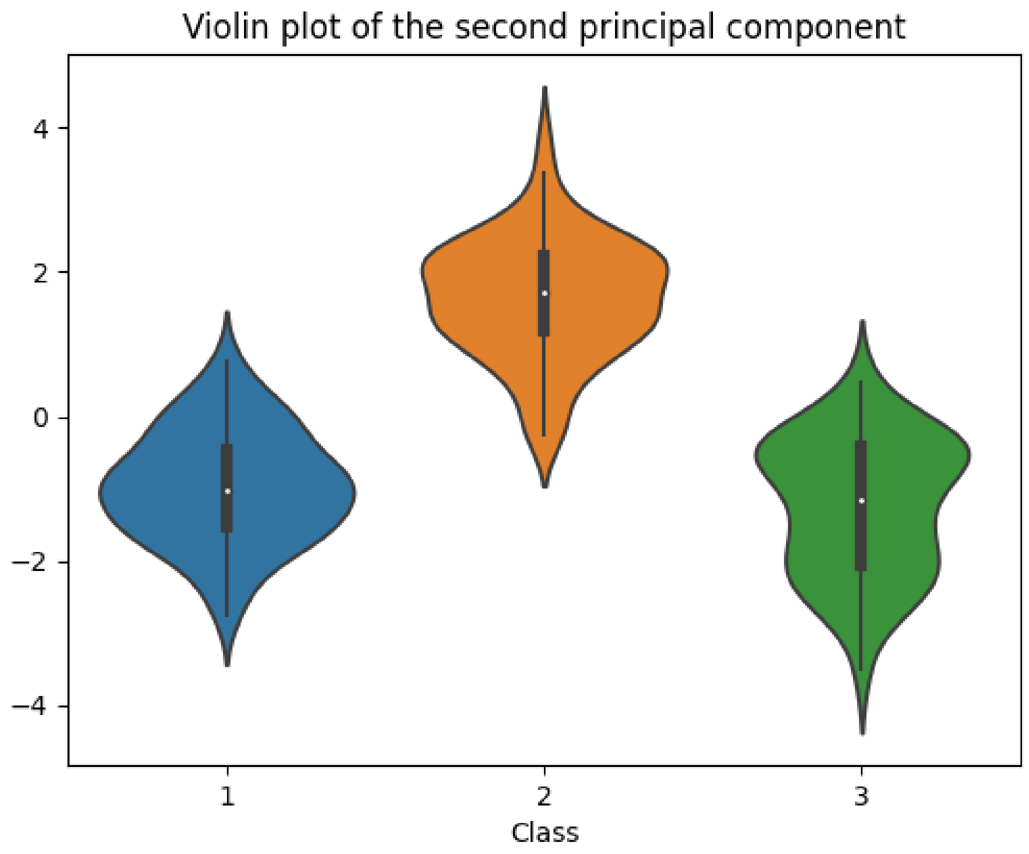
Violion plots

[236]:
```
# Create a violin plot of the first principal component
sns.violinplot(x=y, y=X_pca[:, 0])

plt.title('Violin plot of the first principal component')
plt.show()
```



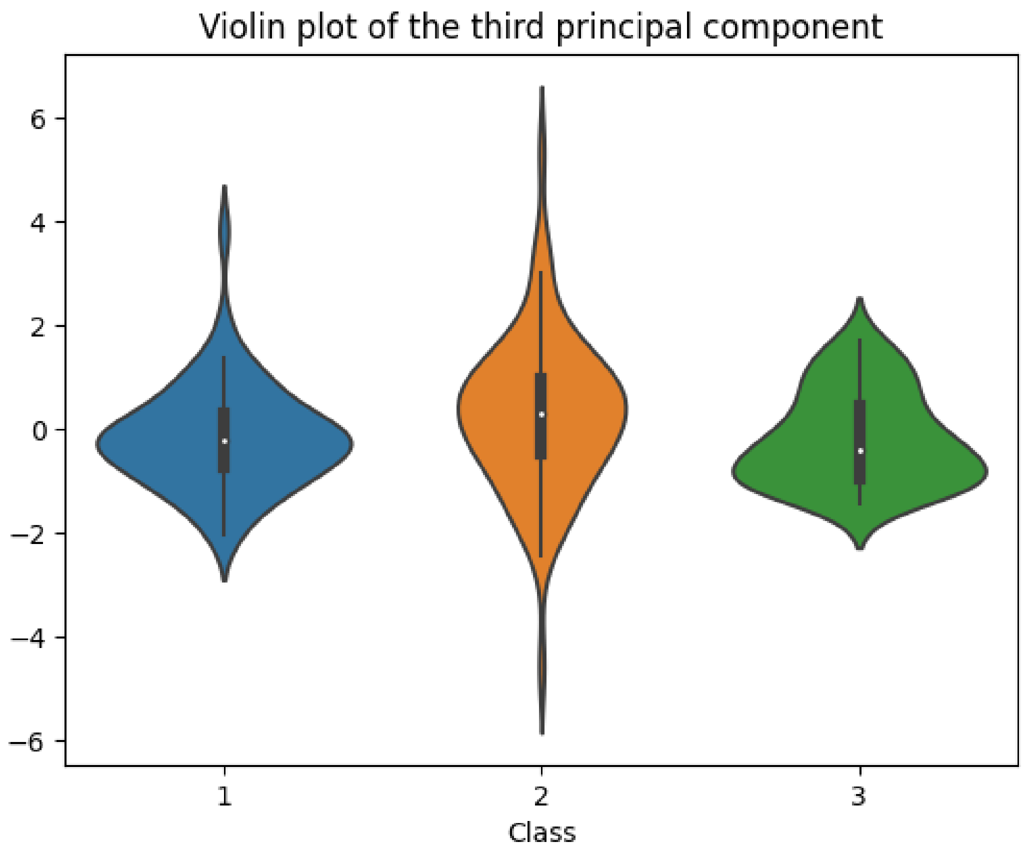
```
[247]: fig = px.violin(title='Interactive Violin plot of the first principal␣
         ↪component', x=y, y=X_pca[:,0])
       fig.show()
```

```
[216]: # Create a violin plot of the second principal component
       sns.violinplot(x=y, y=X_pca[:, 1])

       plt.title('Violin plot of the second principal component')
       plt.show()
```



[248]:
```
fig = px.violin(title='Interactive Violin plot of the second principal␣
↪component', x=y, y=X_pca[:,1])
fig.show()
```

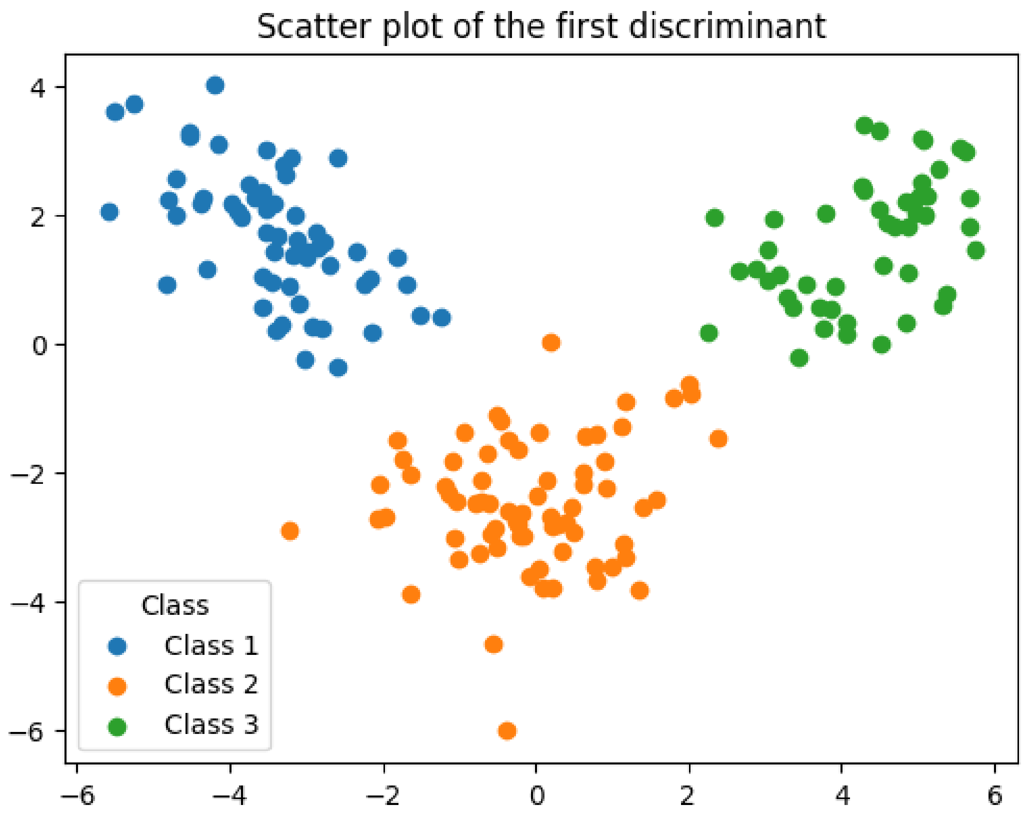
[217]:
```
# Create a violin plot of the third principal component
sns.violinplot(x=y, y=X_pca[:, 2])

plt.title('Violin plot of the third principal component')
plt.show()
```



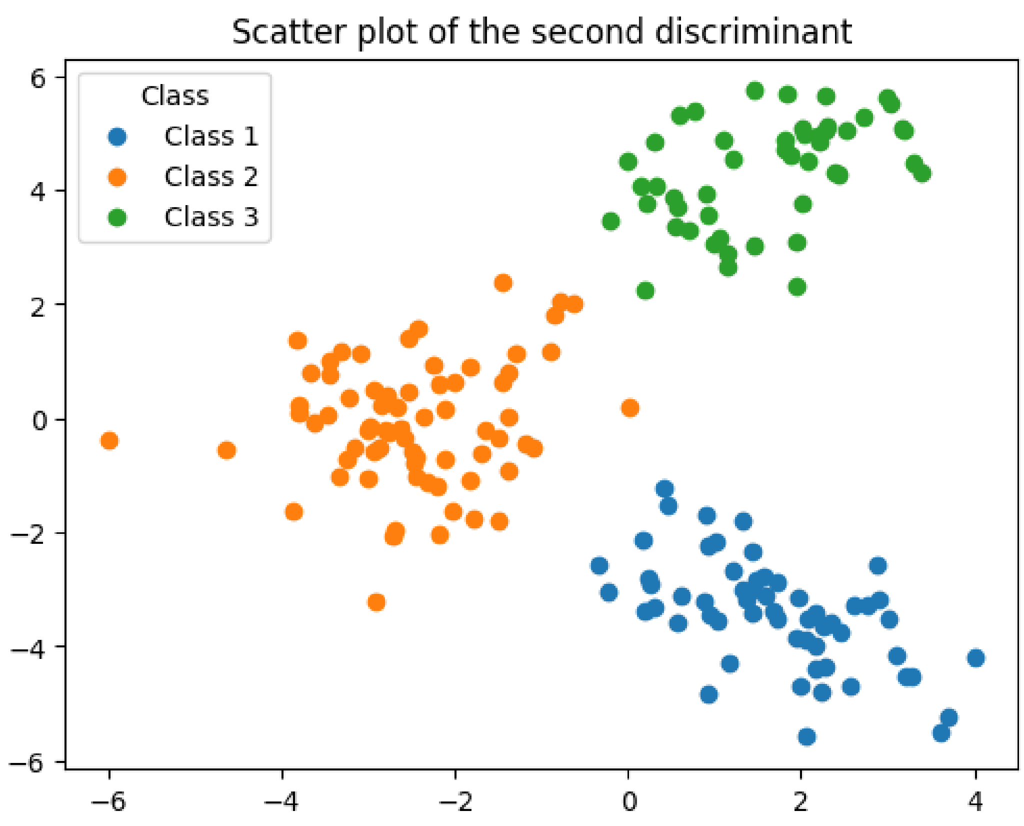
[249]:
```
fig = px.violin(title='Interactive Violin plot of the third principal␣
  ↪component', x=y, y=X_pca[:,2])
fig.show()
```

Linear Discriminant Analysis Perform Linear Discriminant Analysis using standard scaler

[231]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

lda = LinearDiscriminantAnalysis(n_components=2)

lda.fit(X_scaled, y)

X_lda = lda.transform(X_scaled)

for label in np.unique(y):
    mask = y == label
    plt.scatter(X_lda[mask, 0], X_lda[mask, 1], label=f'Class {label}')
plt.legend(title='Class')


plt.title('Scatter plot of the first discriminant')
plt.show()
```

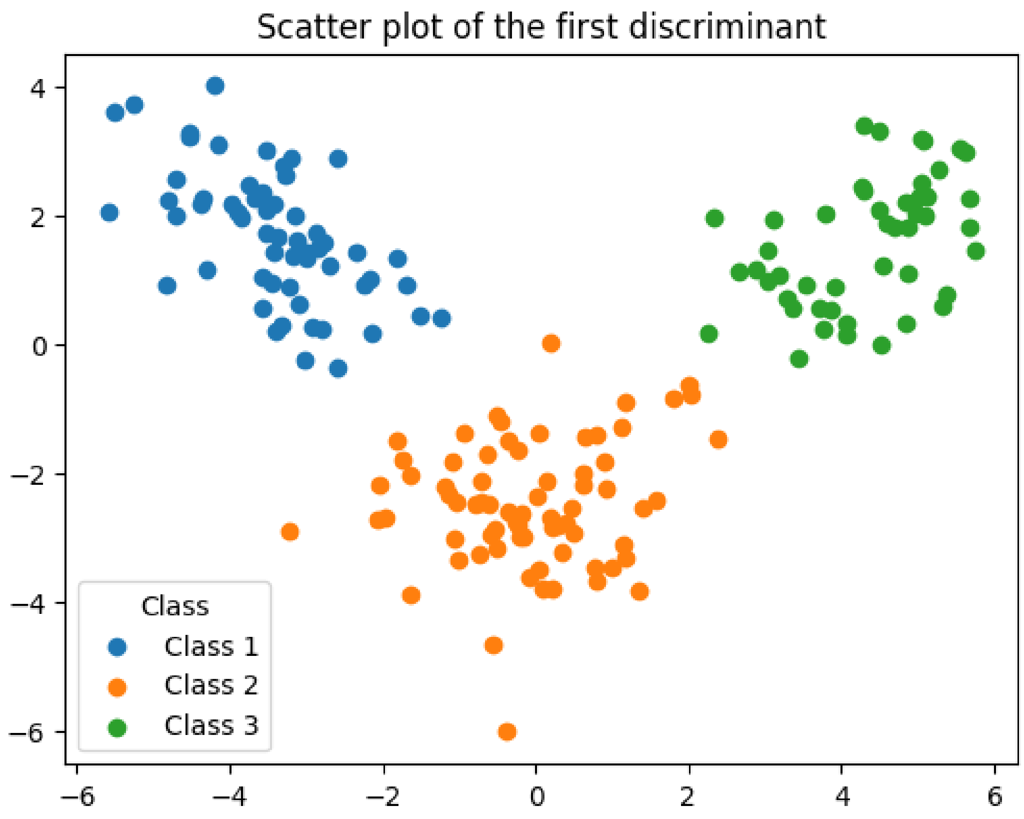


[232]:
```
for label in np.unique(y):
    mask = y == label
    plt.scatter(X_lda[mask, 1], X_lda[mask, 0], label=f'Class {label}')

plt.legend(title='Class')


plt.title('Scatter plot of the second discriminant')
plt.show()
```

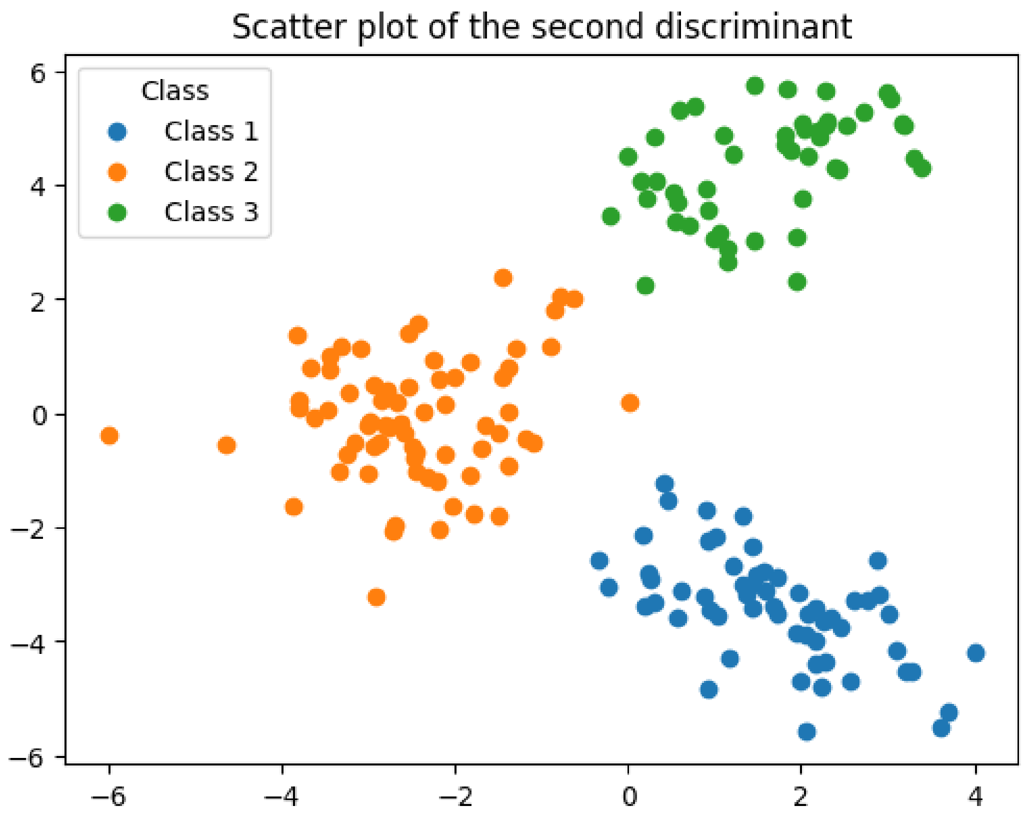


Perform Linear Discriminant Analysis using min-max scaler

[233]:
```
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

lda = LinearDiscriminantAnalysis(n_components=2)

lda.fit(X_scaled, y)

X_lda = lda.transform(X_scaled)

for label in np.unique(y):
    mask = y == label
    plt.scatter(X_lda[mask, 0], X_lda[mask, 1], label=f'Class {label}')

plt.legend(title='Class')


plt.title('Scatter plot of the first discriminant')
plt.show()
```



[234]:
```
for label in np.unique(y):
    mask = y == label
    plt.scatter(X_lda[mask, 1], X_lda[mask, 0], label=f'Class {label}')

plt.legend(title='Class')


plt.title('Scatter plot of the second discriminant')
plt.show()
```


Scatter plot of the second discriminant
Class
Class 1
Class 2
Class 3
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4