K-Means-clustering-and-PCA-predictions-by-retzam-ai

July 21, 2024

[3]:
```
# We'll import some needed packages like panda, numpy...
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

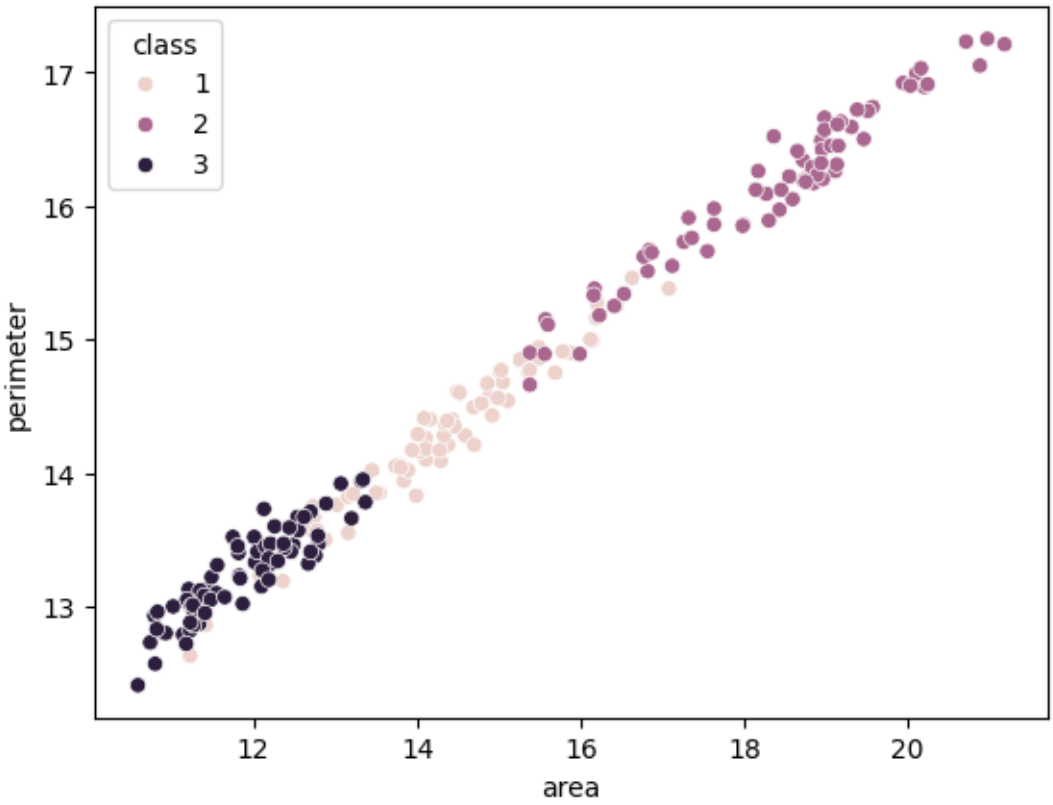
[4]:
```
# Create cols headers to use in reading dataset.
cols = ["area", "perimeter", "compactness", "length", "width", "asymmetry",␣
 ↪"groove", "class"]

# Import dataset
df = pd.read_csv("seeds_dataset.txt", names=cols, sep="\s+")
```

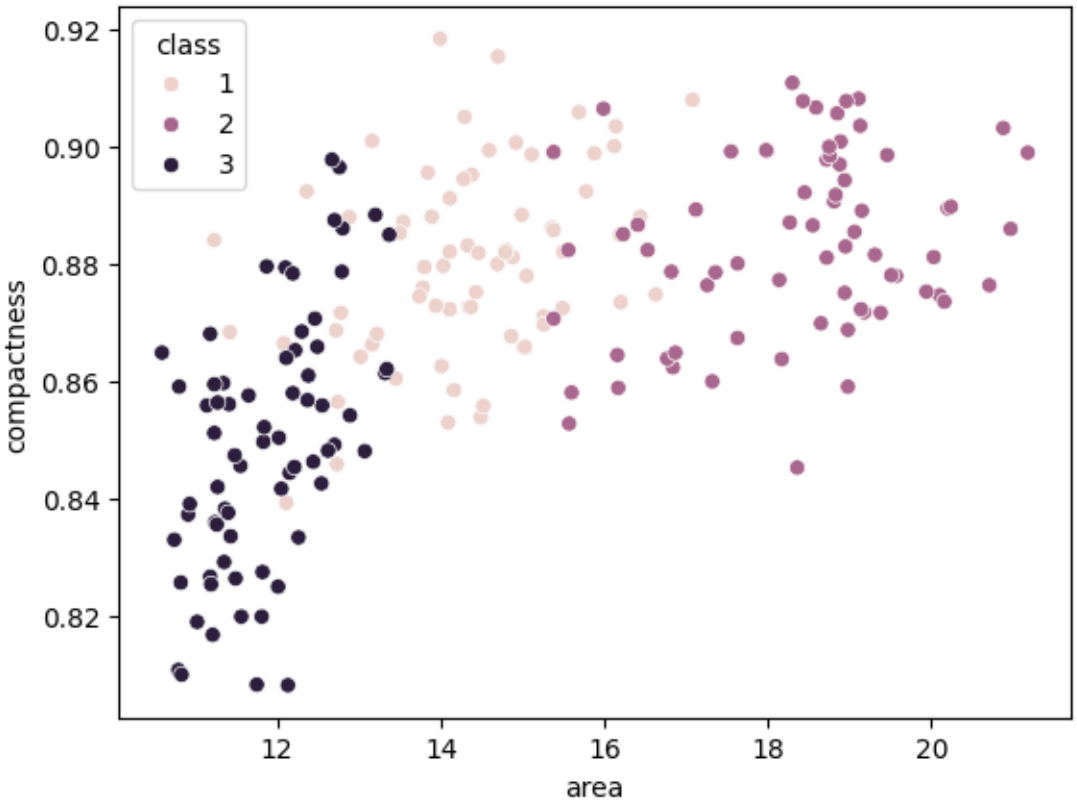
[5]:
```
df.head()
```

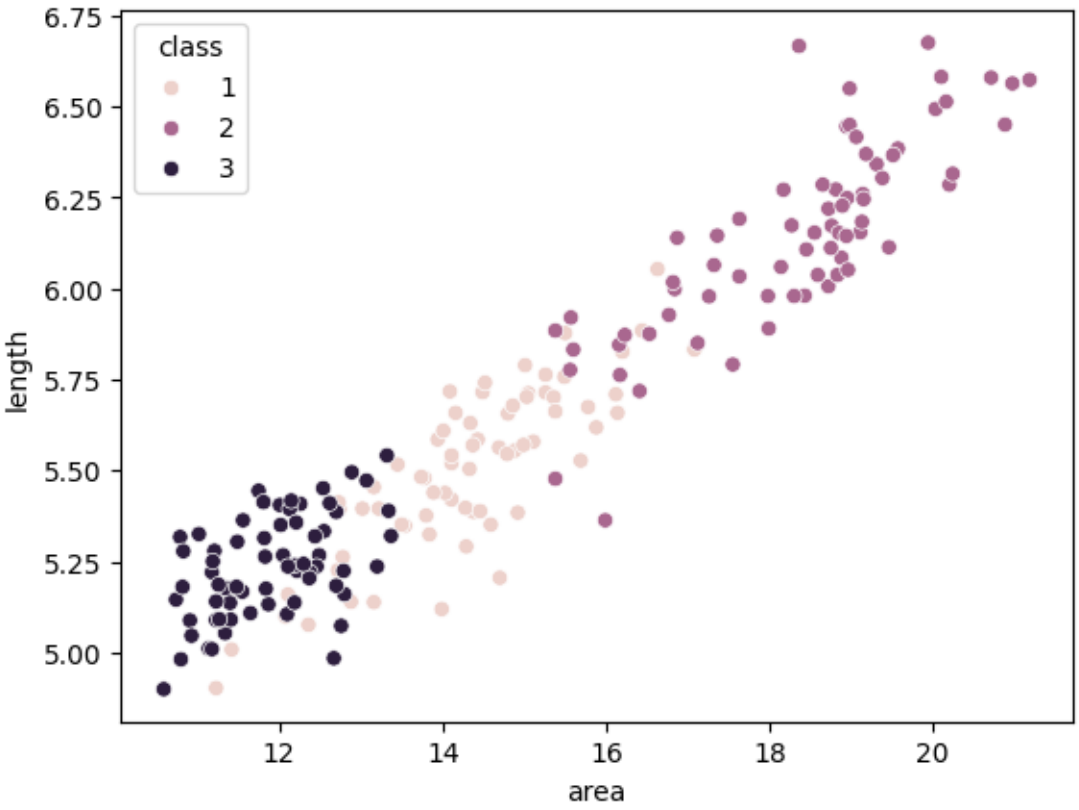
[5]:

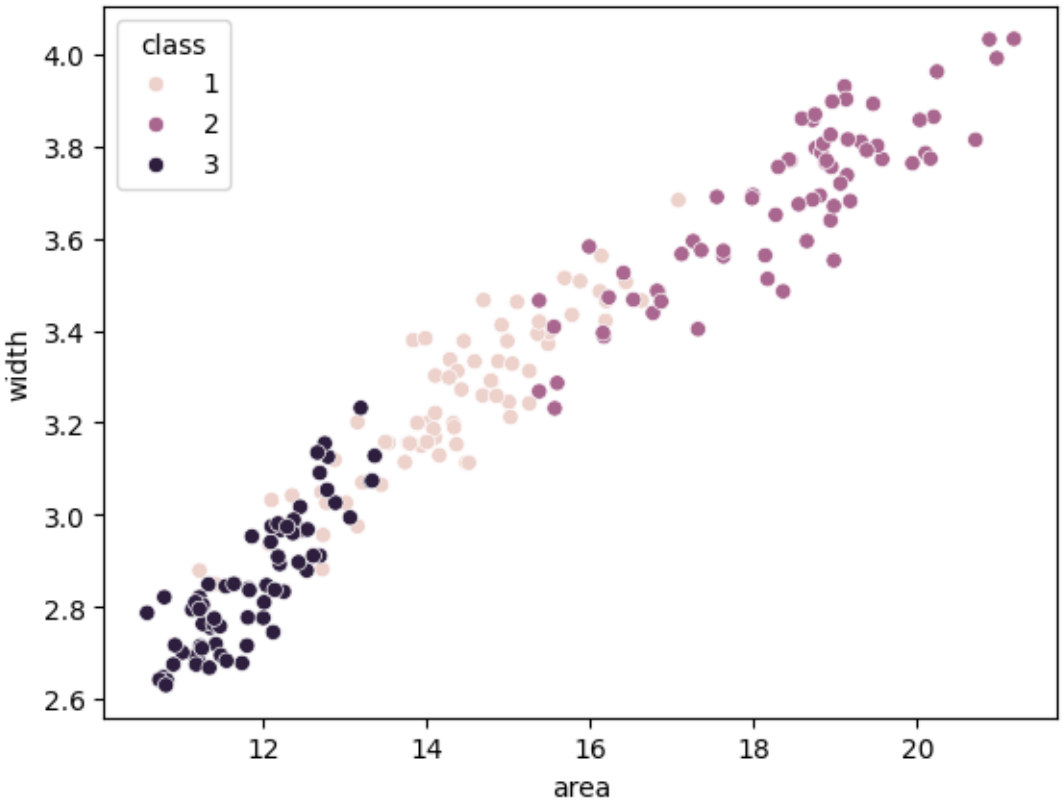
	area	perimeter	compactness	length	width	asymmetry	groove	class
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220	1
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	1
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825	1
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	1
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	1

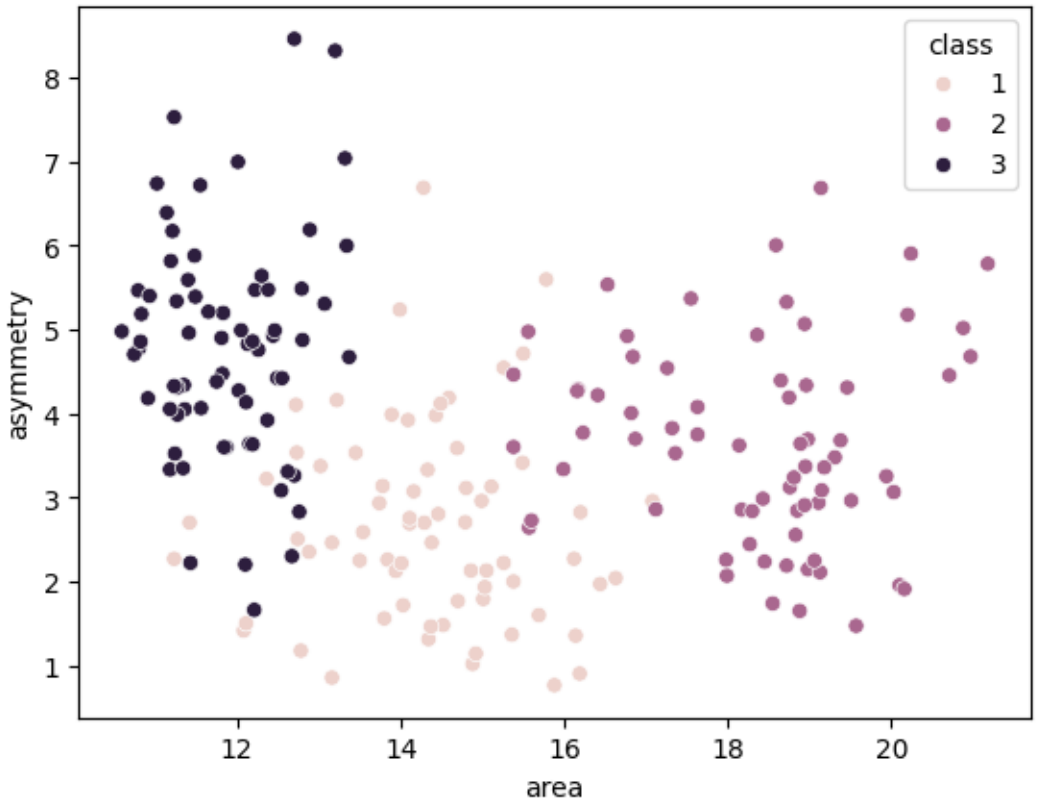
[6]:
```
# Create a scatter plot of all features to the class
# This would help us see the different patterns for each
# feature in relation to the class it belongs to.
for i in range(len(cols)-1):
  for j in range(i+1, len(cols)-1):
    x_label = cols[i]
    y_label = cols[j]
    sns.scatterplot(x=x_label, y=y_label, data=df, hue="class")
    plt.show()
```

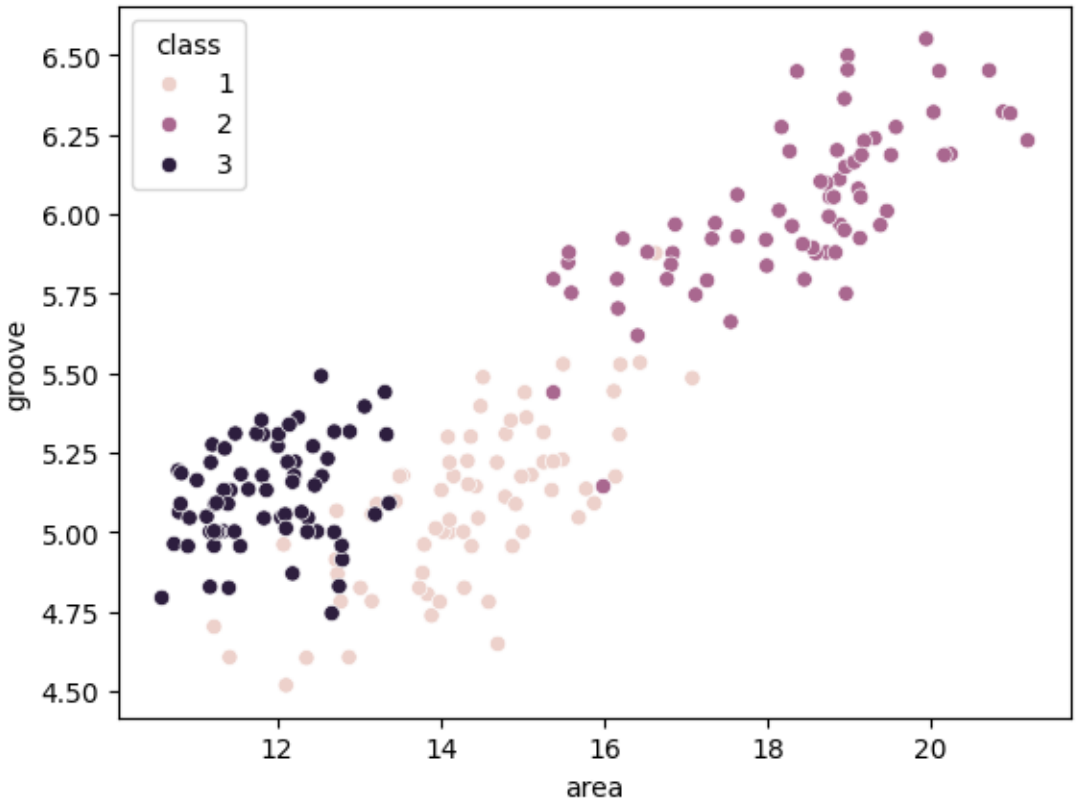

class
1
2
3
perimeter
17
16
15
14
13
area
12
14
16
18
20

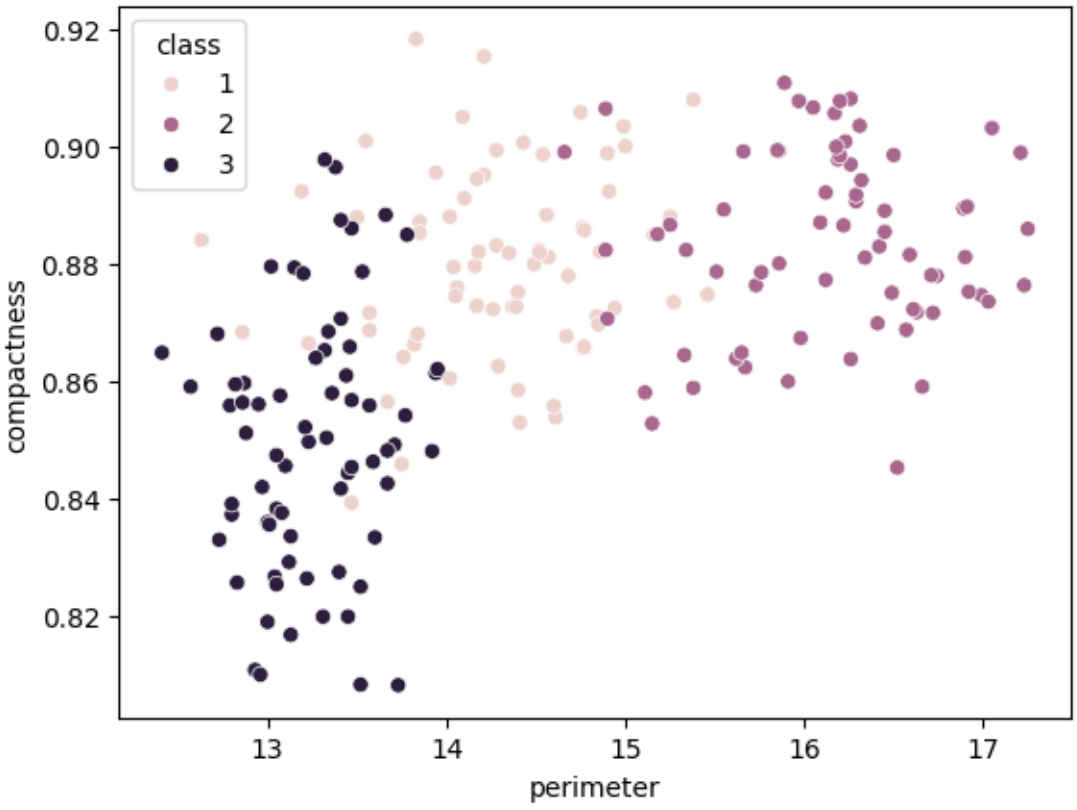

class
1
2
3
0.92
0.90
0.88
0.86
0.84
0.82
compactness
12
14
16
18
20
area

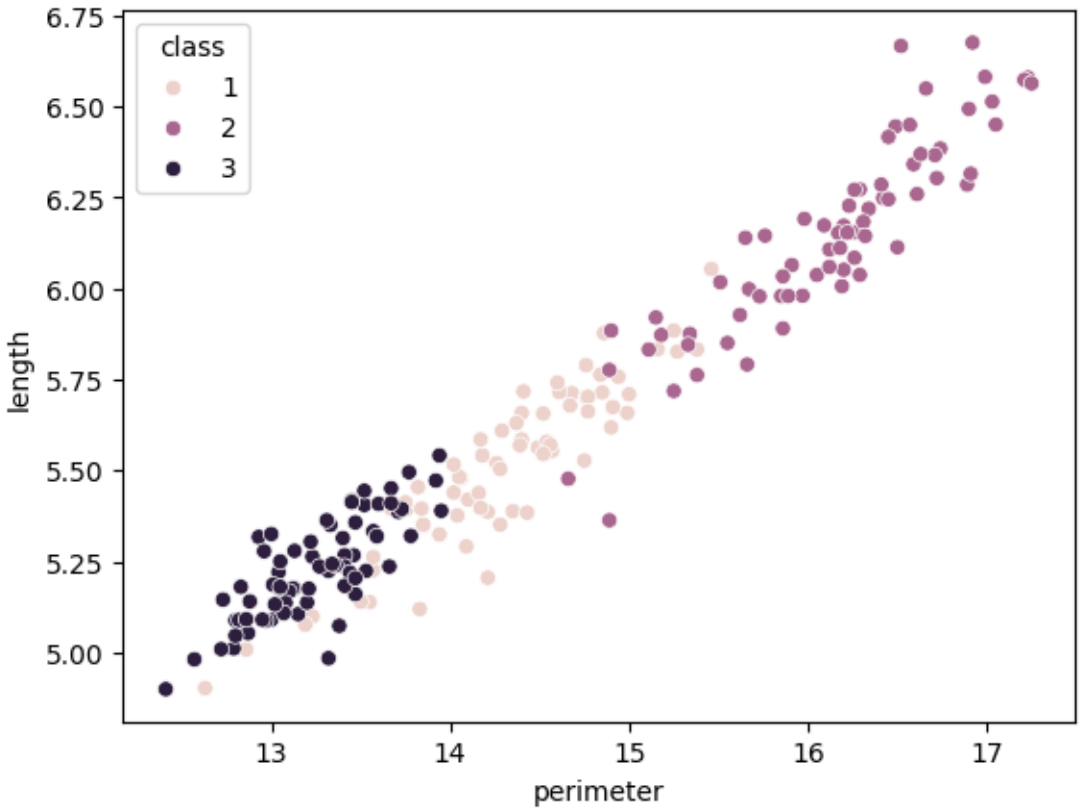

class
1
2
3
6.75
6.50
6.25
6.00
5.75
5.50
5.25
5.00
length
12
14
16
18
20
area

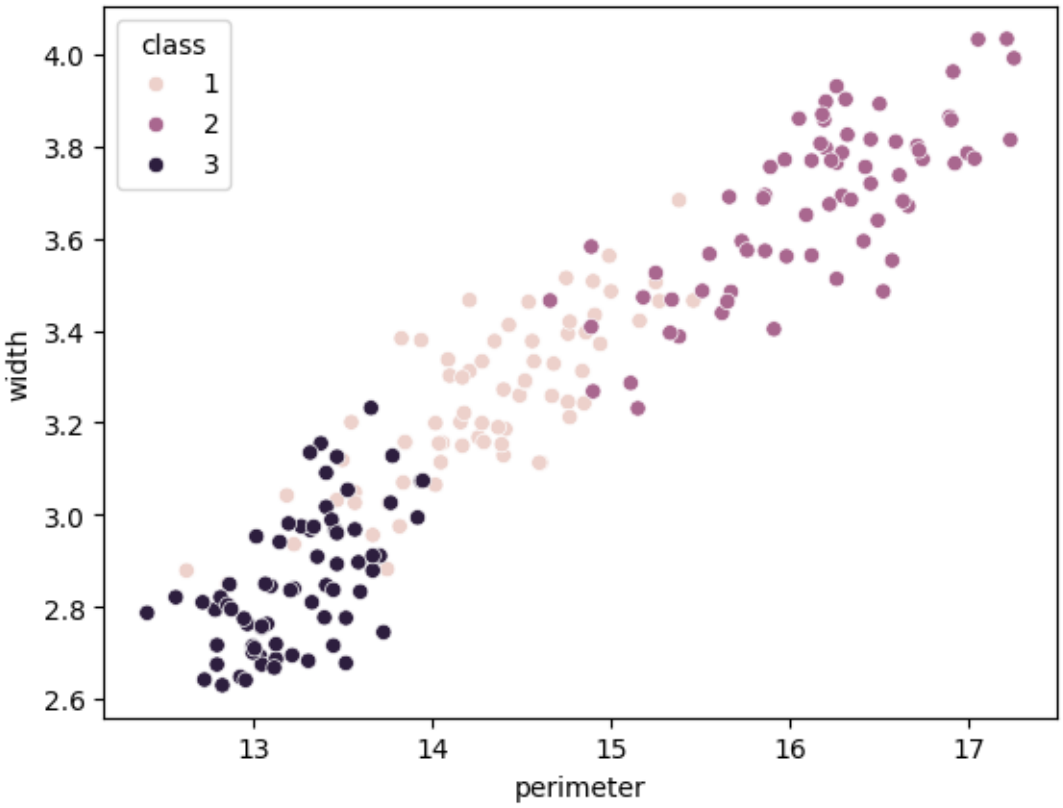

class
1
2
3
4.0
3.8
3.6
3.4
3.2
3.0
2.8
2.6
width
12
14
16
18
20
area

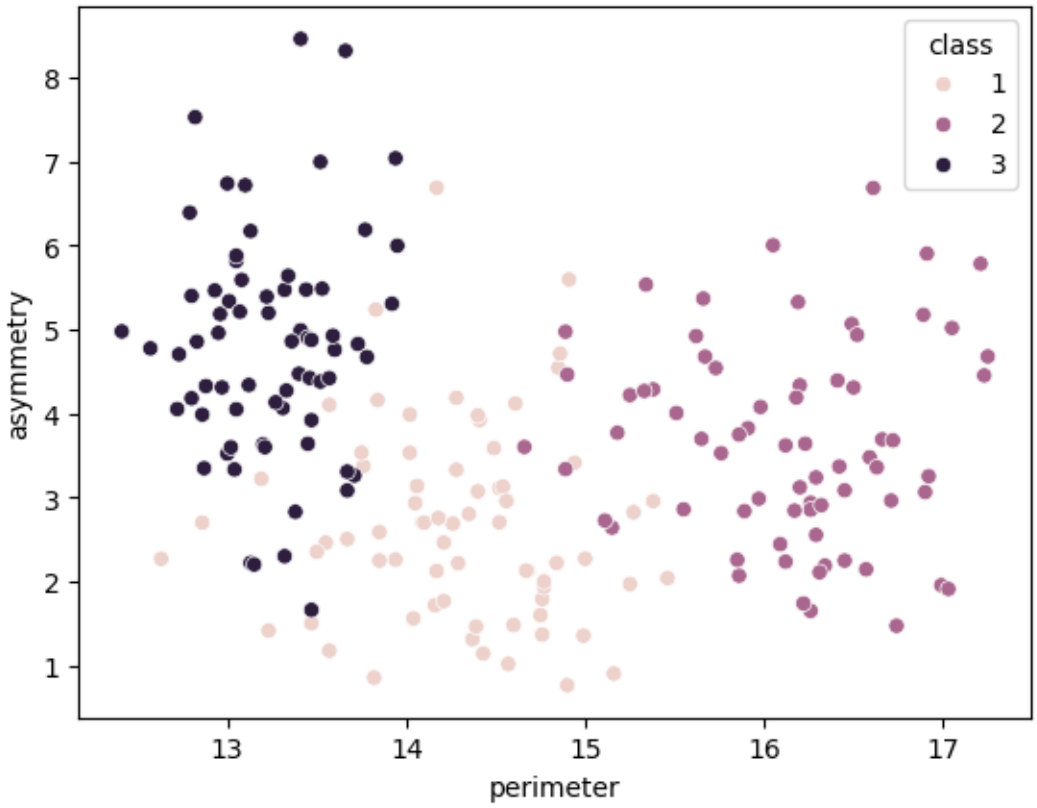

class
1
2
3
asymmetry
8
7
6
5
4
3
2
1
12
14
16
18
20
area

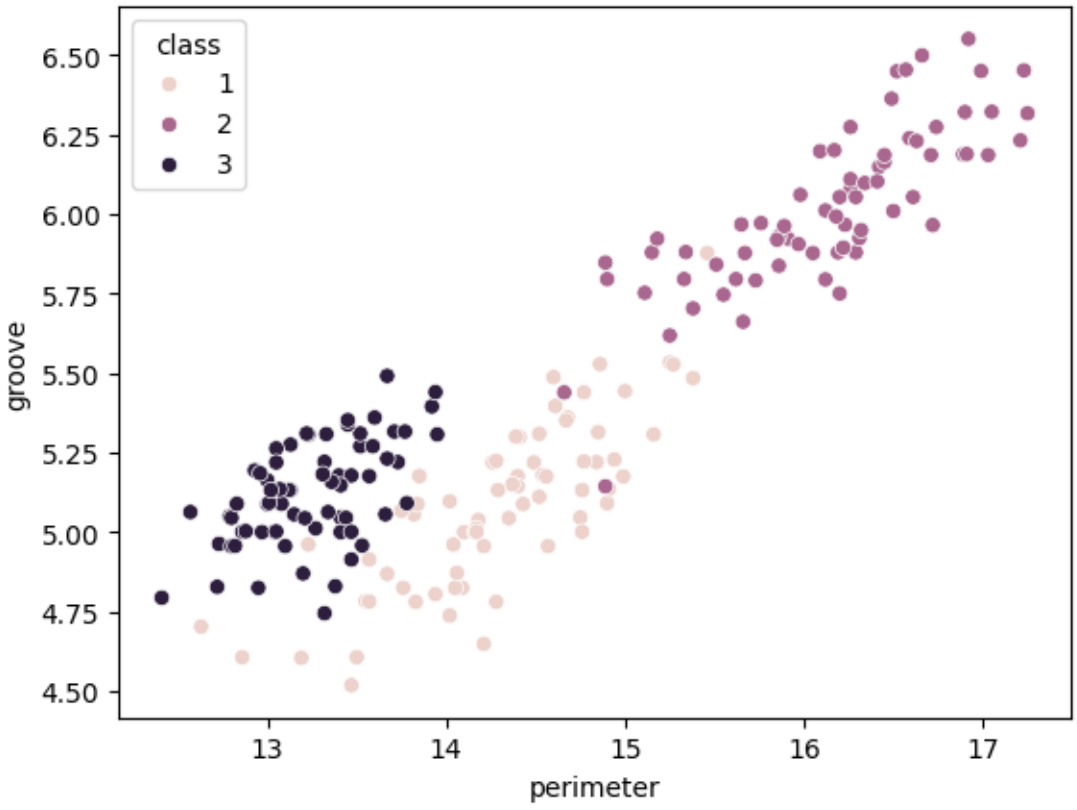

class
1
2
3
groove
area
6.50
6.25
6.00
5.75
5.50
5.25
5.00
4.75
4.50
12
14
16
18
20

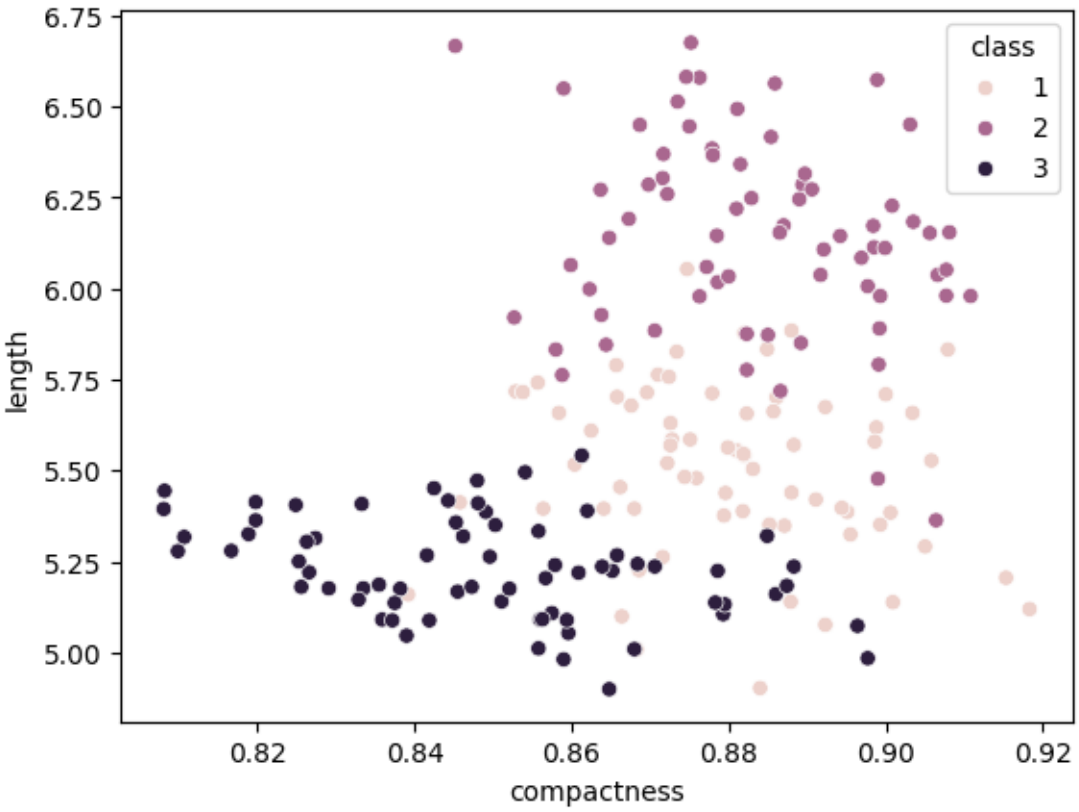

class
1
2
3
0.92
0.90
0.88
0.86
0.84
0.82
compactness
13
14
15
16
17
perimeter

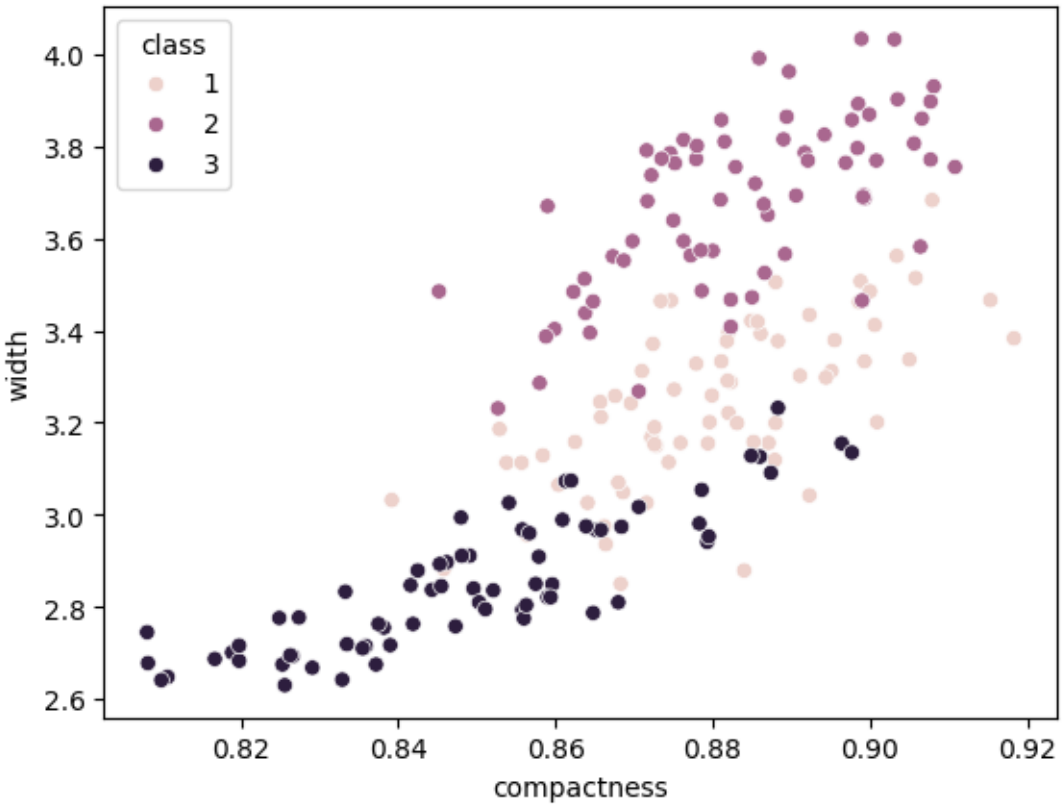

class
1
2
3
6.75
6.50
6.25
6.00
5.75
5.50
5.25
5.00
length
13
14
15
16
17
perimeter

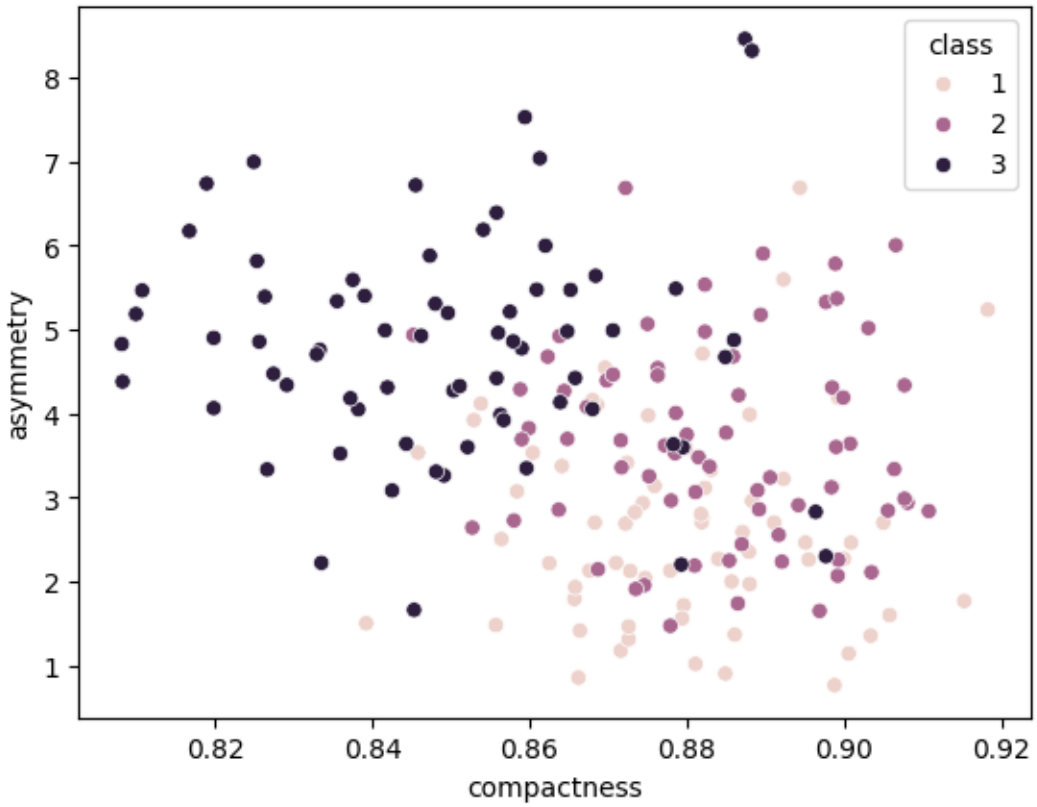

class
1
2
3
4.0
3.8
3.6
3.4
3.2
3.0
2.8
2.6
width
13
14
15
16
17
perimeter

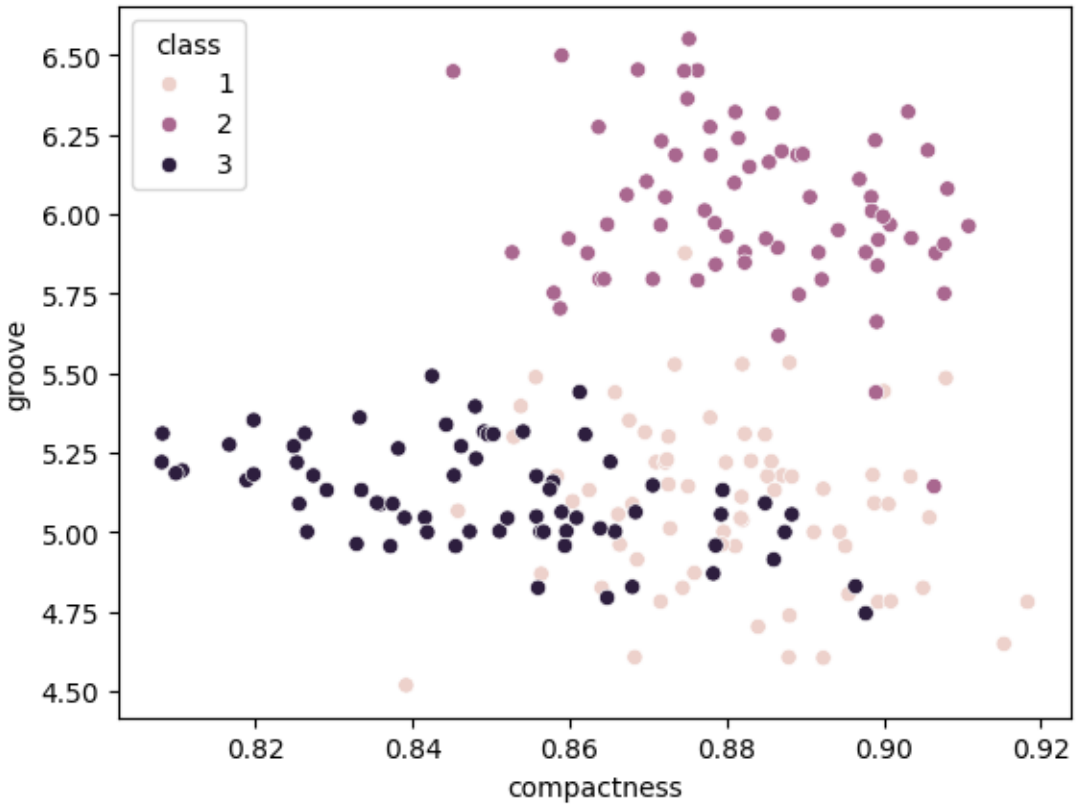

class
1
2
3
asymmetry
perimeter
8
7
6
5
4
3
2
1
13
14
15
16
17

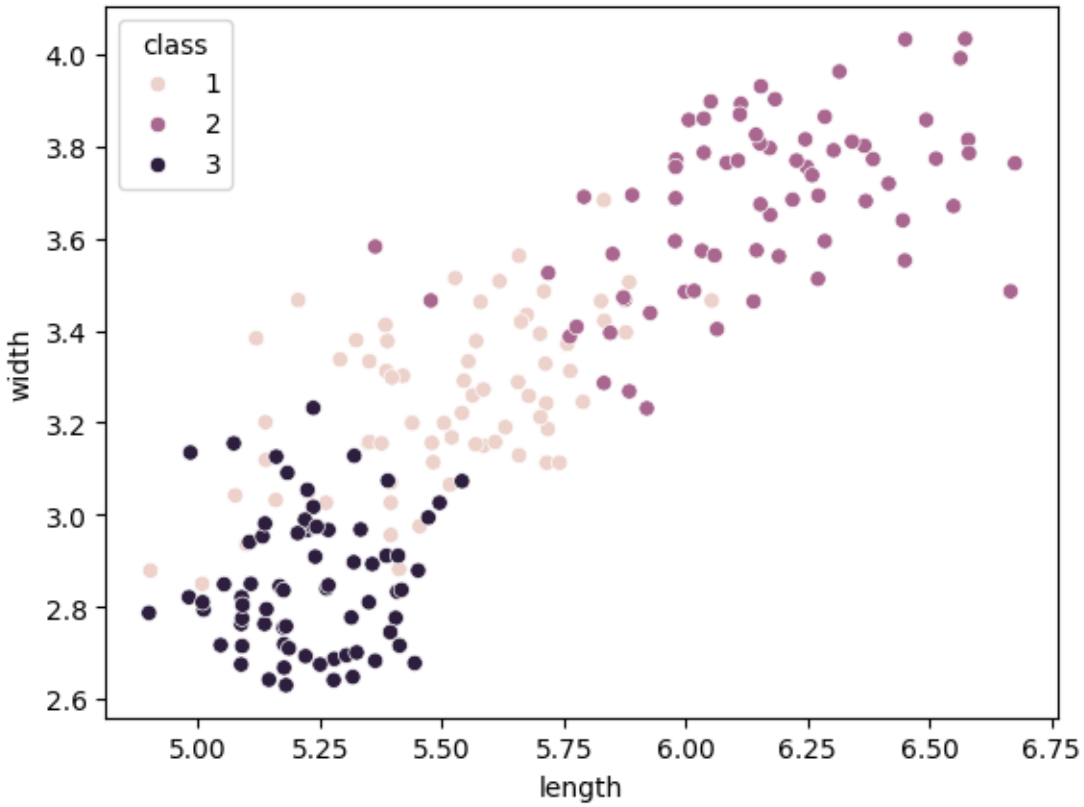

class
1
2
3
6.50
6.25
6.00
5.75
5.50
5.25
5.00
4.75
4.50
groove
13
14
15
16
17
perimeter

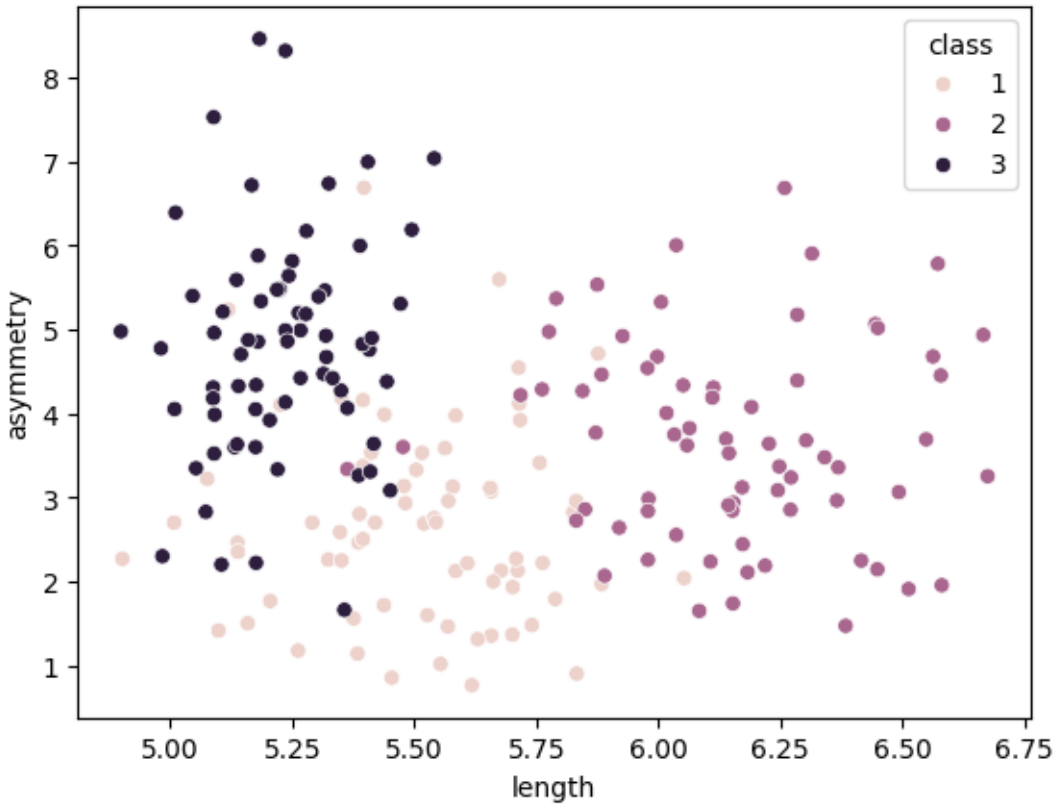

class
1
2
3
length
compactness
6.75
6.50
6.25
6.00
5.75
5.50
5.25
5.00
0.82
0.84
0.86
0.88
0.90
0.92

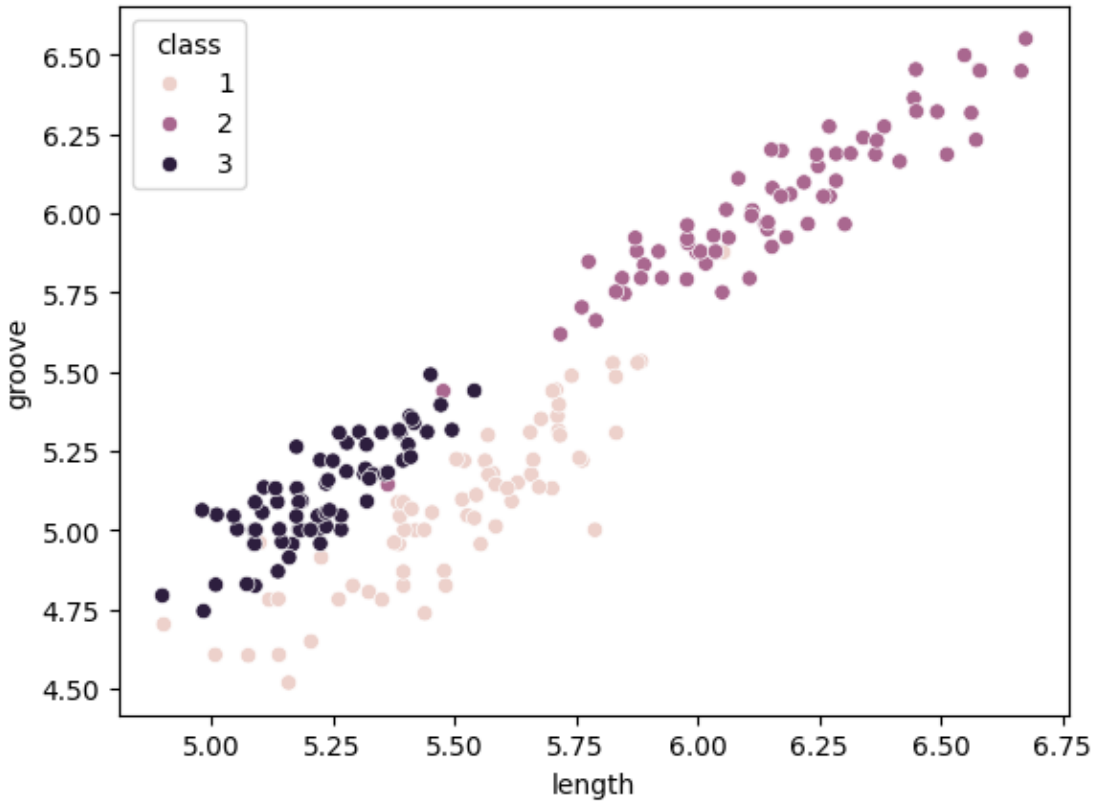

class
1
2
3
4.0
3.8
3.6
3.4
3.2
3.0
2.8
2.6
width
0.82
0.84
0.86
0.88
0.90
0.92
compactness

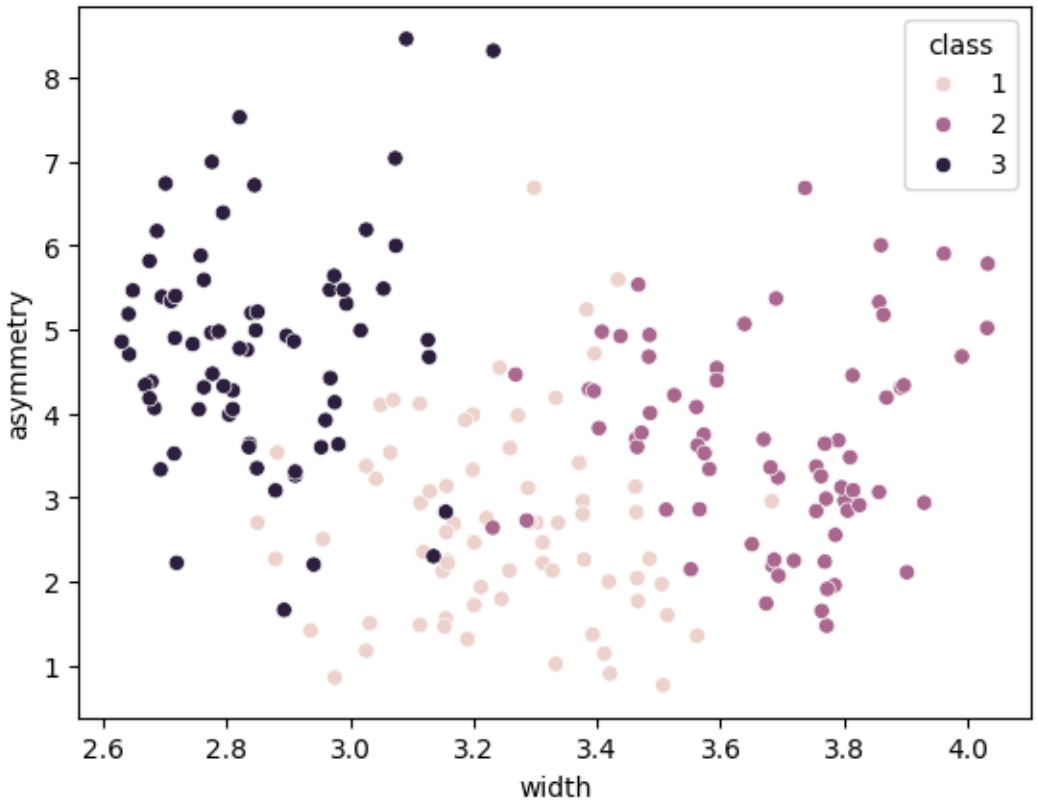

class
1
2
3
asymmetry
compactness
8
7
6
5
4
3
2
1
0.82
0.84
0.86
0.88
0.90
0.92

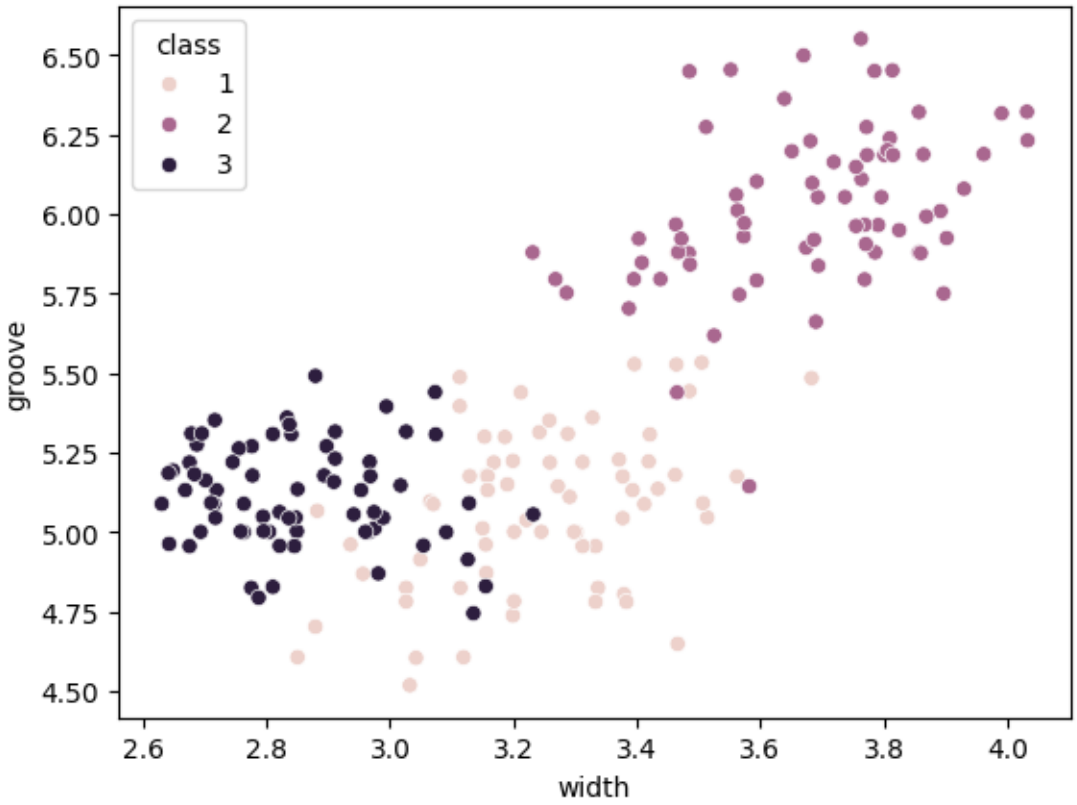

class
1
2
3
groove
compactness
6.50
6.25
6.00
5.75
5.50
5.25
5.00
4.75
4.50
0.82
0.84
0.86
0.88
0.90
0.92

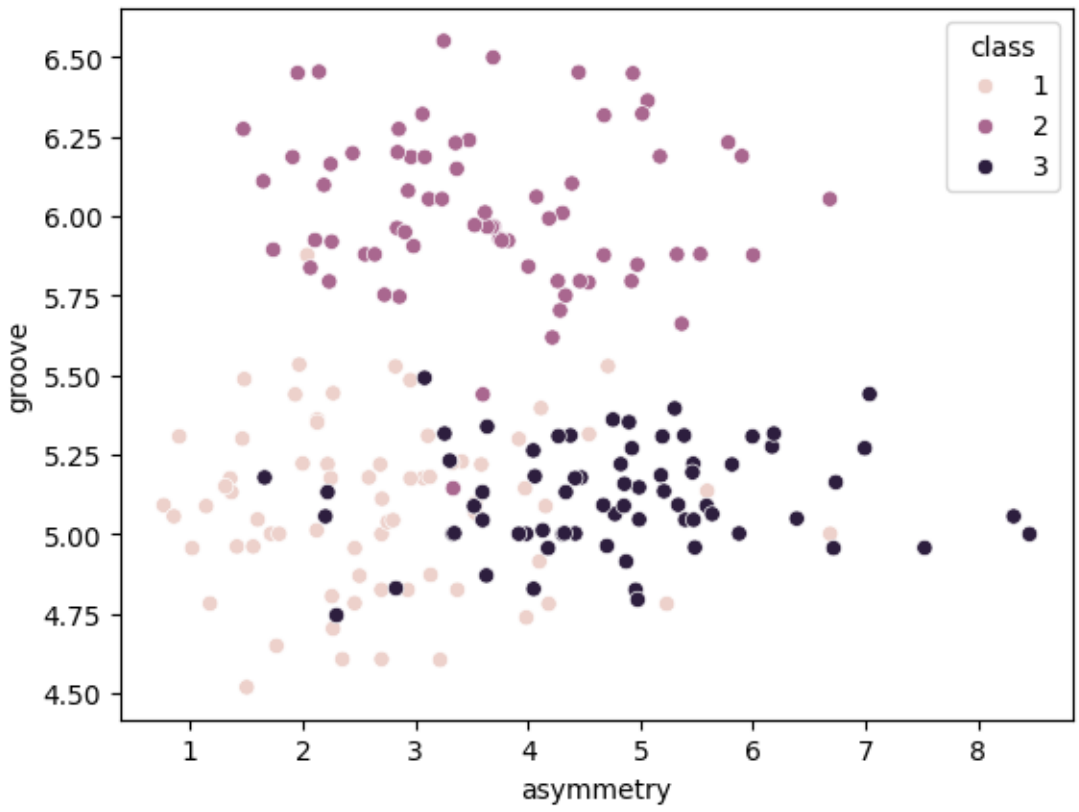

class
1
2
3
4.0
3.8
3.6
3.4
3.2
3.0
2.8
2.6
width
5.00
5.25
5.50
5.75
6.00
6.25
6.50
6.75
length


class
1
2
3
asymmetry
length
8
7
6
5
4
3
2
1
5.00
5.25
5.50
5.75
6.00
6.25
6.50
6.75


class
1
2
3
groove
length
6.50
6.25
6.00
5.75
5.50
5.25
5.00
4.75
4.50
5.00
5.25
5.50
5.75
6.00
6.25
6.50
6.75


class
1
2
3
asymmetry
width
8
7
6
5
4
3
2
1
2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0


class
1
2
3
groove
6.50
6.25
6.00
5.75
5.50
5.25
5.00
4.75
4.50
2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0
width



[8]:
```
# CLUSTERING
```

[9]:
```
# Lets try to train a model to cluster this dataset with KMeans Clustering.
from sklearn.cluster import KMeans
```

[11]:
```
# We'll use the perimeter and asymmetry features of the dataset
x = "perimeter"
y = "asymmetry"
X = df[[x, y]].values
```

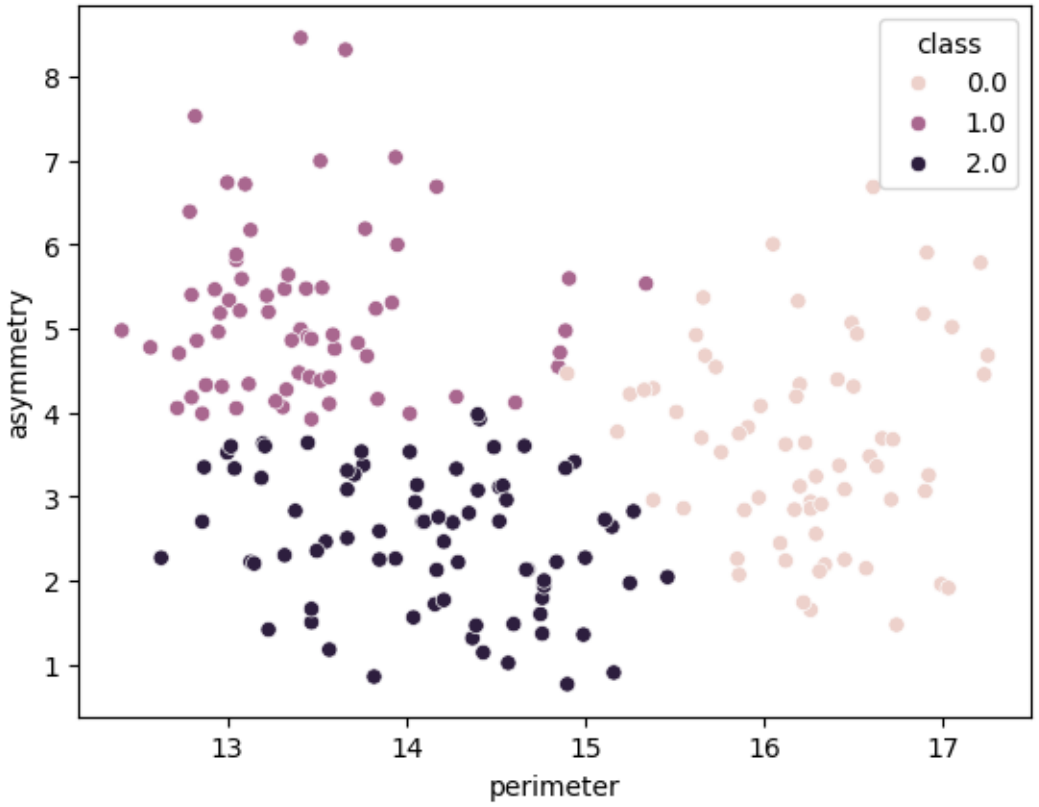
[12]:
```
# This creates a KMeans cluster with K=3, since we have three groups.
kmeans = KMeans(n_clusters = 3, n_init=10).fit(X)
```

[13]:
```
# This creates the predicted clusters for each data item (observation).
clusters = kmeans.labels_
```

[15]:
```
# Create a cluster dataframe to visualize our predictions.
cluster_df = pd.DataFrame(np.hstack((X, clusters.reshape(-1, 1))), columns=[x,␣
↪y, "class"])
```

[16]:
```
# K Means classes
sns.scatterplot(x=x, y=y, hue="class", data=cluster_df)
plt.plot()
```

[16]: []



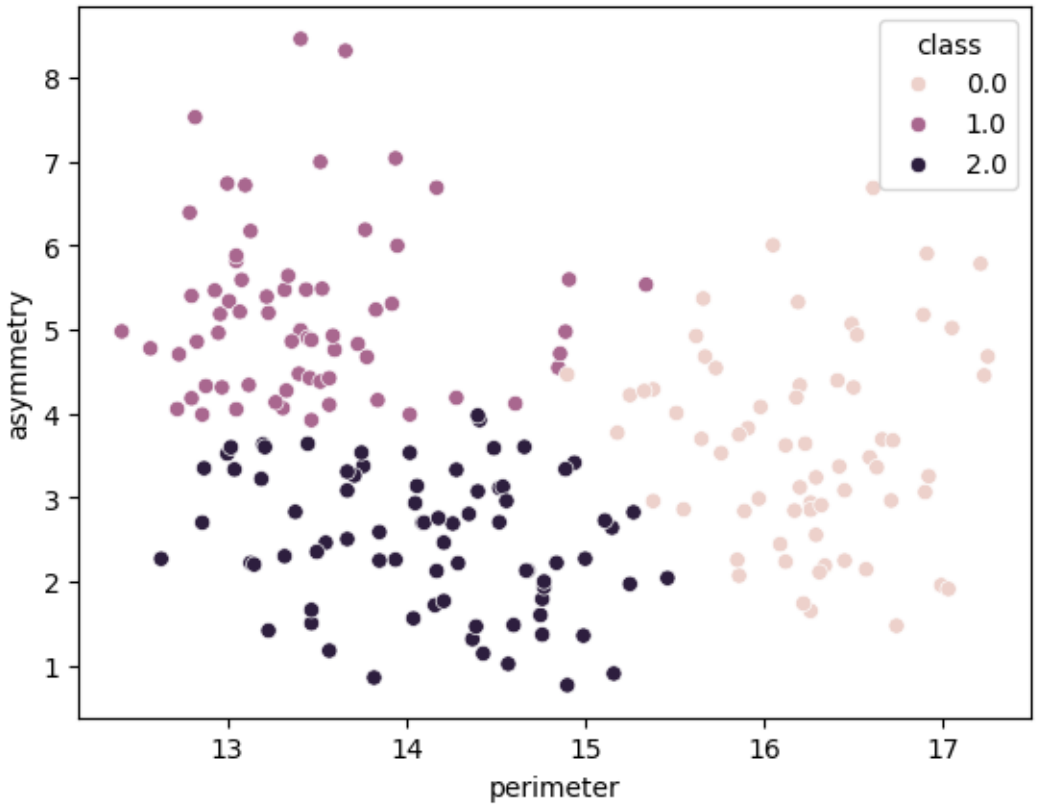
[ ]:
```
# HIGHER DIMENSIONS
```

[17]:
```
# With higher dimensions we want to train our model with all the features
# in the dataset.
X = df[cols[:-1]].values
```

[18]:
```
kmeans = KMeans(n_clusters = 3, n_init=10).fit(X)
cluster_df = pd.DataFrame(np.hstack((X, clusters.reshape(-1, 1))), columns=df.
 ↪columns)
```

[19]:
```
# K Means classes
sns.scatterplot(x=x, y=y, hue="class", data=cluster_df)
plt.plot()
```

[19]: []



[ ]:
```
# PRINCIPAL COMPONENT ANALYSIS (PCA)
```

[20]:
```
# With PCA we are reducing dimensions.
from sklearn.decomposition import PCA
```

[21]:
```
# Create PCA with 2 dimensions.
pca = PCA(n_components=2)

# Train PCA
transformed_x = pca.fit_transform(X)
```

[22]:
```
# Our X showing (210, 7), means 210 samples, with 7 dimensions(features)
X.shape
```

[22]: (210, 7)

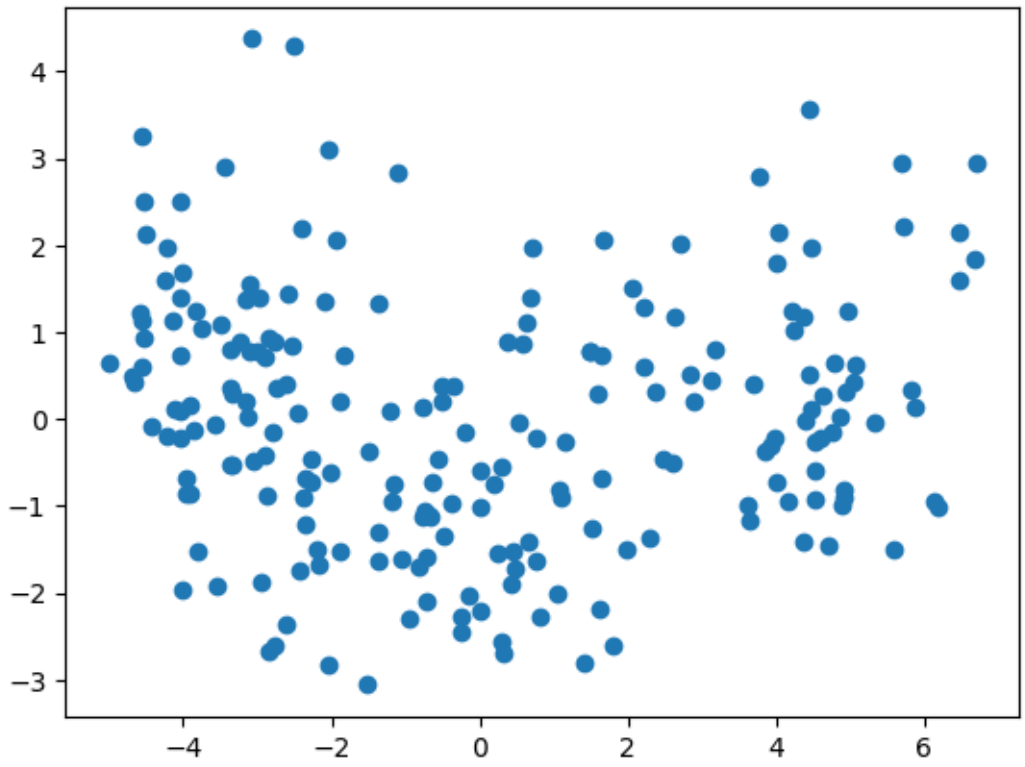
[23]:
```
# Our transformed X showing (210, 2),
# means 210 samples, with 2 dimensions(features)
transformed_x.shape
```

[23]: (210, 2)

[24]:
```
# Printing transformed_x, shows our 2 dimensions.
transformed_x[:5]
```

[24]:
```
array([[ 0.66344838, -1.41732098],
       [ 0.31566651, -2.68922915],
       [-0.6604993 , -1.13150635],
       [-1.0552759 , -1.62119002],
       [ 1.61999921, -2.18338442]])
```

[25]:
```
# We can plot this to visualize the 2 dimensions of transformed_x
plt.scatter(transformed_x[:,0], transformed_x[:, 1])
plt.show()
```



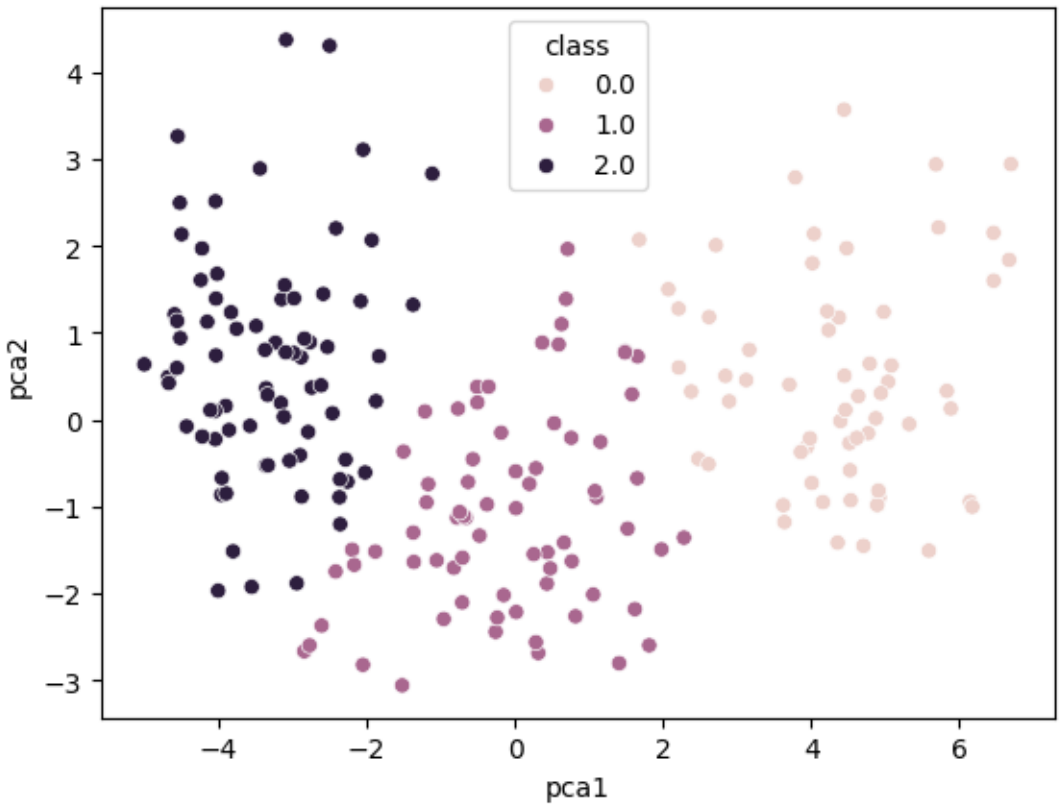
[26]:
```
# Now we can use our transformed_x dataset, to make predictions
# with KMeans clustering, and we specify the 2 PCA dimensions:
# pca1 and pca2
kmeans_pca_df = pd.DataFrame(np.hstack((transformed_x, kmeans.labels_.
 ↪reshape(-1, 1))), columns=["pca1", "pca2", "class"])
```

[27]:
```
# We want to compare kmeans_pca_df which is the predicted values
# to truth_pca_df with the original values in the dataset to see
# our accuracy
```

```
truth_pca_df = pd.DataFrame(np.hstack((transformed_x, kmeans.labels_.
 ↪reshape(-1, 1))), columns=["pca1", "pca2", "class"])
```

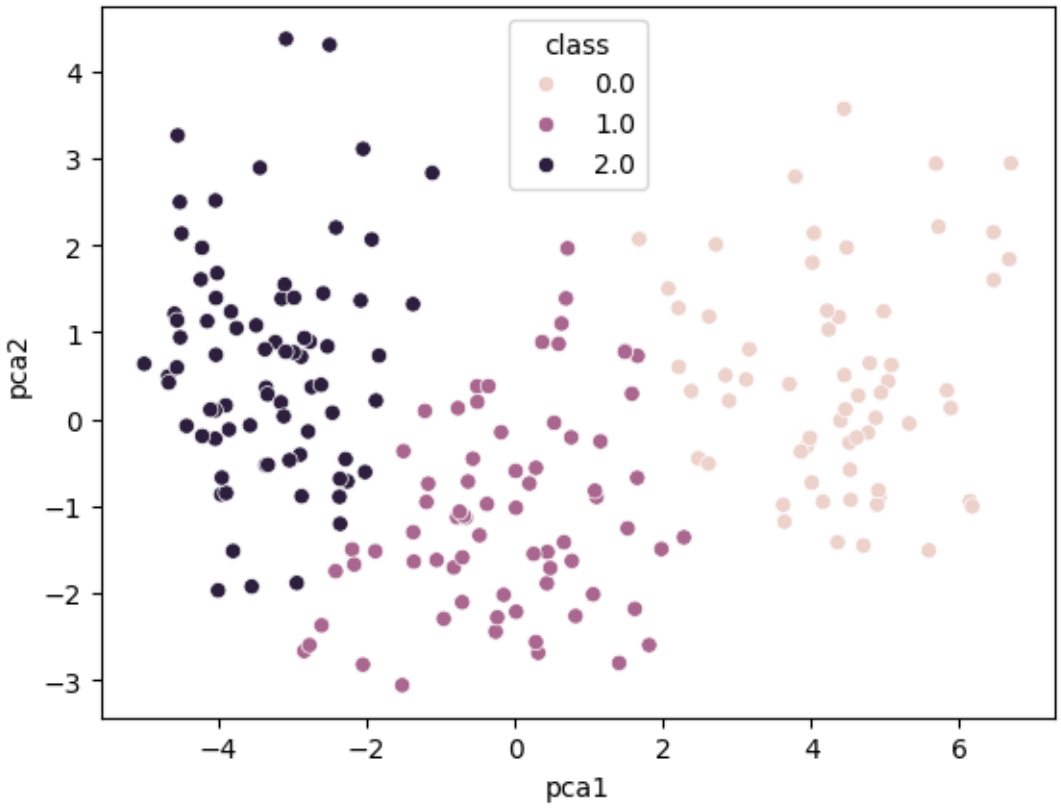
[28]:
```
# K Means classes
sns.scatterplot(x="pca1", y="pca2", hue="class", data=kmeans_pca_df)
plt.plot()
```

[28]: []



[29]:
```
# Truth classes
sns.scatterplot(x="pca1", y="pca2", hue="class", data=truth_pca_df)
plt.plot()
```

[29]: []


class
0.0
1.0
2.0
pca2
pca1
4
3
2
1
0
−1
−2
−3
−4
−2
0
2
4
6

[ ]: