

# Labeled Data

Separate the PCA components of a dataset by class.

## Chapter Goals:

- Learn about labeled datasets
- Separate principle component data by class label

### A. Class labels

A big part of data science is classifying observations in a dataset into separate categories, or *classes*. A popular use case of data classification is in separating a dataset into "good" and "bad" categories. For example, we can classify a dataset of breast tumors as either malignant or benign.

The code below separates a breast cancer dataset into malignant and benign categories. The `load_breast_cancer` ([https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load\\_breast\\_cancer.html#sklearn.datasets.load\\_breast\\_cancer](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html#sklearn.datasets.load_breast_cancer)) function is part of the scikit-learn library, and its data comes from the Breast Cancer Wisconsin (Diagnostic) dataset ([https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Diagnostic\)](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic))).

```
1 from sklearn.datasets import load_breast_cancer
2 bc = load_breast_cancer()
3 print('{}\n'.format(repr(bc.data)))
4 print('Data shape: {}\n'.format(bc.data.shape))
5
6 # Class labels
7 print('{}\n'.format(repr(bc.target)))
8 print('Labels shape: {}\n'.format(bc.target.shape))
9
10 # Label names
11 print('{}\n'.format(list(bc.target_names)))
12
13 malignant = bc.data[bc.target == 0]
14 print('Malignant shape: {}\n'.format(malignant.shape))
15
16 benign = bc.data[bc.target == 1]
17 print('Benign shape: {}\n'.format(benign.shape))
```



In the example above, the `bc.data` array contains all the dataset values, while the `bc.target` array contains the class ID labels for each row in `bc.data`. A class ID of 0 corresponds to a malignant tumor, while a class ID of 1 corresponds to a benign tumor.

Using the `bc.target` class IDs, we separated the dataset into malignant and benign data arrays. In other words, the `malignant` array contains the rows of `bc.data` corresponding to the indexes in `bc.target` containing 0, while the `benign` array contains the rows of `bc.data` corresponding to the indexes in `bc.target` containing 1. There are 212 malignant data observations and 357 benign observations.

## Time to Code!

The coding exercise in this chapter involves completing the `separate_components` function, which will separate principal component data by class.

To do this, we first need to complete a helper function, `get_label_info`, which returns the label name and data for an input class label.

The `component_data` input represents the principal component data.



The `labels` input is a 1-D array containing the class label IDs corresponding to each row of `component_data`. We can use it to separate the principle components by class.

The `class_label` input represents a particular class label ID.

The `label_names` input represents all the string names for the class labels.

**Set `label_name` equal to the string at index `class_label` of `label_names`.**

**Set `label_data` equal to the rows of `component_data` corresponding to the indexes where `labels` equals `class_label`. Then return the tuple `(label_name, label_data)`.**

```
1 def get_label_info(component_data, labels,
2                     class_label, label_names):
3     # CODE HERE
4     pass
```



Solution

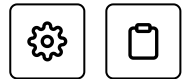


```
1 def get_label_info(component_data, labels,
2                     class_label, label_names):
3     label_name = label_names[class_label]
4     label_data = component_data[labels == class_label]
5     return (label_name, label_data)
6
7
```

Now, inside the main `separate_data` function, we'll iterate through each label in the `label_names` list.

**Set `separated_data` equal to an empty list.**

Create a `for` loop that iterates variable `class_label` through `range(len(label_names))`.



Inside the `for` loop, we can use our helper function to obtain the separated data for each class.

Inside the `for` loop, call `get_label_info` with inputs `component_data`, `labels`, `class_label`, and `label_names`. Append the function's output to `separated_data`.

After finalizing the list of principle components separated by class, we return it.

Outside the `for` loop, return `separated_data`.

```
1 def separate_data(component_data, labels,
2                     label_names):
3     # CODE HERE
4     pass
5
```



Solution



```
1 def separate_data(component_data, labels,
2                     label_names):
3     separated_data = []
4     for class_label in range(len(label_names)):
5         separated_data.append(get_label_info(
6             component_data, labels, class_label, label_names))
7     return separated_data
8
9
```

The `separate_data` function is incredibly useful for visualizing the components. We can use `matplotlib` to create nice plots of the separated data (shown in the code below).

```
1 from sklearn.datasets import load_breast_cancer
```

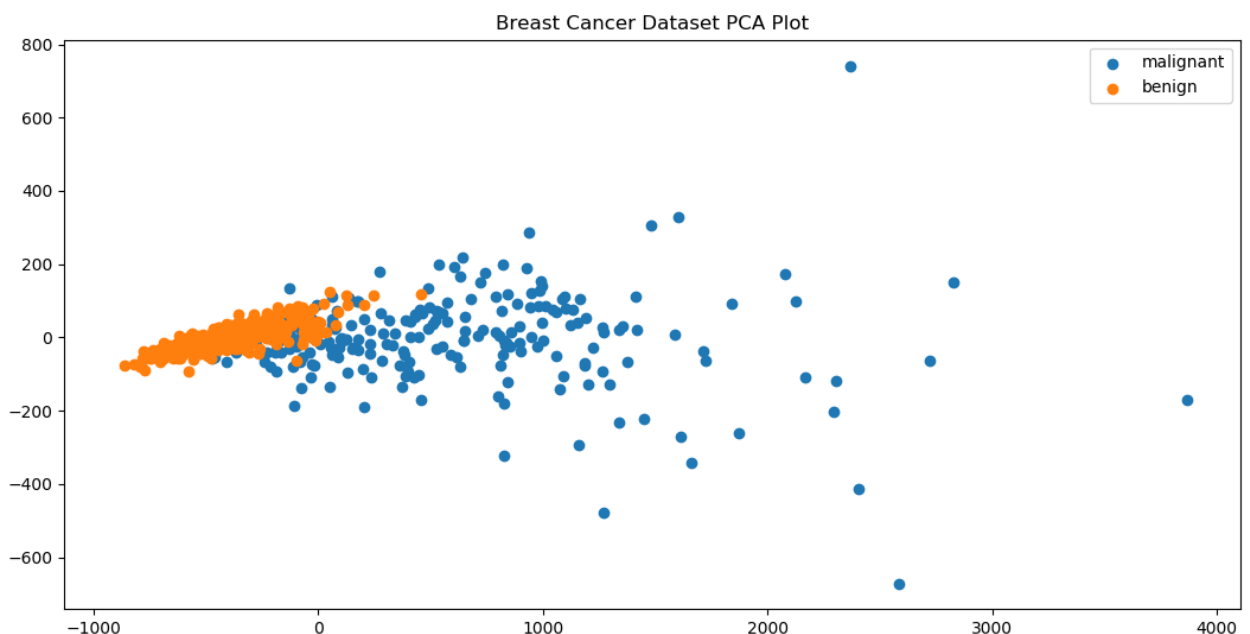


```
2 from sklearn.decomposition import PCA
3 bc = load_breast_cancer()
4 pca_obj = PCA(n_components=2)
5 component_data = pca_obj.fit_transform(bc.data)
6 labels = bc.target
7 label_names = bc.target_names
8 # Using the completed separate_data function
9 separated_data = separate_data(component_data,
10                                labels, label_names)
11
12 # Plotting the data
13 import matplotlib.pyplot as plt
14 for label_name, label_data in separated_data:
15     col1 = label_data[:, 0] # 1st column (1st pr. comp.)
16     col2 = label_data[:, 1] # 2nd column (2nd pr. comp.)
17     plt.scatter(col1, col2, label=label_name) # scatterplot
18 plt.legend() # adds legend to plot
19 plt.title('Breast Cancer Dataset PCA Plot')
20 plt.show()
```



✓ Succeeded

The result of the above code is this plot:



← Back

PCA

Next



Quiz

☒ Mark as Completed

37% completed, meet the criteria and claim your course certificate!

Buy Certificate



Report  
an Issue



Ask a Question

([https://discuss.educative.io/tag/labeled-data\\_\\_data-preprocessing-with-scikit-learn\\_\\_machine-learning-for-software-engineers](https://discuss.educative.io/tag/labeled-data__data-preprocessing-with-scikit-learn__machine-learning-for-software-engineers))