Now we can instantiate and run the grid search as usual, here on the `cancer` dataset:

In[36]:

```
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, random_state=0)

grid = GridSearchCV(pipe, param_grid, cv=5)
grid.fit(X_train, y_train)

print("Best params:\n{}\n".format(grid.best_params_))
print("Best cross-validation score: {:.2f}".format(grid.best_score_))
print("Test-set score: {:.2f}".format(grid.score(X_test, y_test)))
```

Out[36]:

```
Best params:
{'classifier':
 SVC(C=10, cache_size=200, class_weight=None, coef0=0.0,
     decision_function_shape=None, degree=3, gamma=0.01, kernel='rbf',
     max_iter=-1, probability=False, random_state=None, shrinking=True,
     tol=0.001, verbose=False),
 'preprocessing':
 StandardScaler(copy=True, with_mean=True, with_std=True),
 'classifier__C': 10, 'classifier__gamma': 0.01}

Best cross-validation score: 0.99
Test-set score: 0.98
```

The outcome of the grid search is that `SVC` with `StandardScaler` preprocessing, `C=10`, and `gamma=0.01` gave the best result.

Summary and Outlook

In this chapter we introduced the `Pipeline` class, a general-purpose tool to chain together multiple processing steps in a machine learning workflow. Real-world applications of machine learning rarely involve an isolated use of a model, and instead are a sequence of processing steps. Using pipelines allows us to encapsulate multiple steps into a single Python object that adheres to the familiar `scikit-learn` interface of `fit`, `predict`, and `transform`. In particular when doing model evaluation using cross-validation and parameter selection using grid search, using the `Pipeline` class to capture all the processing steps is essential for proper evaluation. The `Pipeline` class also allows writing more succinct code, and reduces the likelihood of mistakes that can happen when building processing chains without the `pipeline` class (like forgetting to apply all transformers on the test set, or not applying them in the right order). Choosing the right combination of feature extraction, preprocessing, and models is somewhat of an art, and often requires some trial and error. However, using pipelines, this "trying out" of many different processing steps is quite simple. When experimenting, be careful not to overcomplicate your processes, and make sure to evaluate whether every component you are including in your model is necessary.

With this chapter, we have completed our survey of general-purpose tools and algorithms provided by `scikit-learn`. You now possess all the required skills and know the necessary mechanisms to apply machine learning in practice. In the next chapter, we will dive in more detail into one particular type of data that is commonly seen in practice, and that requires some special expertise to handle correctly: text data.

CHAPTER 7

Working with Text Data

In Chapter 4, we talked about two kinds of features that can represent properties of the data: continuous features that describe a quantity, and categorical features that are items from a fixed list. There is a third kind of feature that can be found in many applications, which is text. For example, if we want to classify an email message as either a legitimate email or spam, the content of the email will certainly contain important information for this classification task. Or maybe we want to learn about the opinion of a politician on the topic of immigration. Here, that individual's speeches or tweets might provide useful information. In customer service, we often want to find out if a message is a complaint or an inquiry. We can use the subject line and content of a message to automatically determine the customer's intent, which allows us to send the message to the appropriate department, or even send a fully automatic reply.

Text data is usually represented as strings, made up of characters. In any of the examples just given, the length of the text data will vary. This feature is clearly very different from the numeric features that we've discussed so far, and we will need to process the data before we can apply our machine learning algorithms to it.

Types of Data Represented as Strings

Before we dive into the processing steps that go into representing text data for machine learning, we want to briefly discuss different kinds of text data that you might encounter. Text is usually just a string in your dataset, but not all string features should be treated as text. A string feature can sometimes represent categorical variables, as we discussed in Chapter 5. There is no way to know how to treat a string feature before looking at the data.