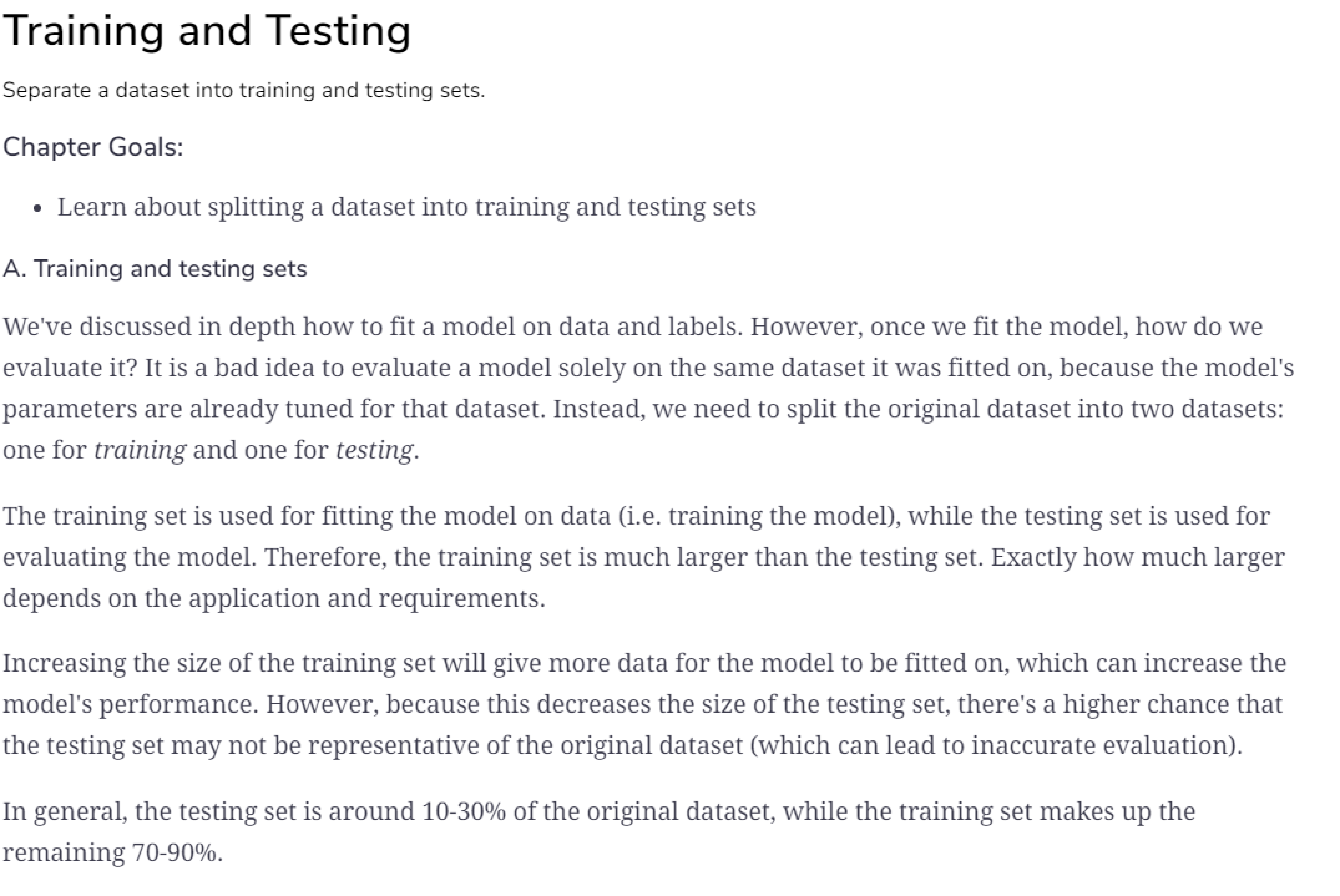


Training and Testing

Separate a dataset into training and testing sets.

Chapter Goals:

- Learn about splitting a dataset into training and testing sets

A. Training and testing sets

We've discussed in depth how to fit a model on data and labels. However, once we fit the model, how do we evaluate it? It is a bad idea to evaluate a model solely on the same dataset it was fitted on, because the model's parameters are already tuned for that dataset. Instead, we need to split the original dataset into two datasets: one for *training* and one for *testing*.

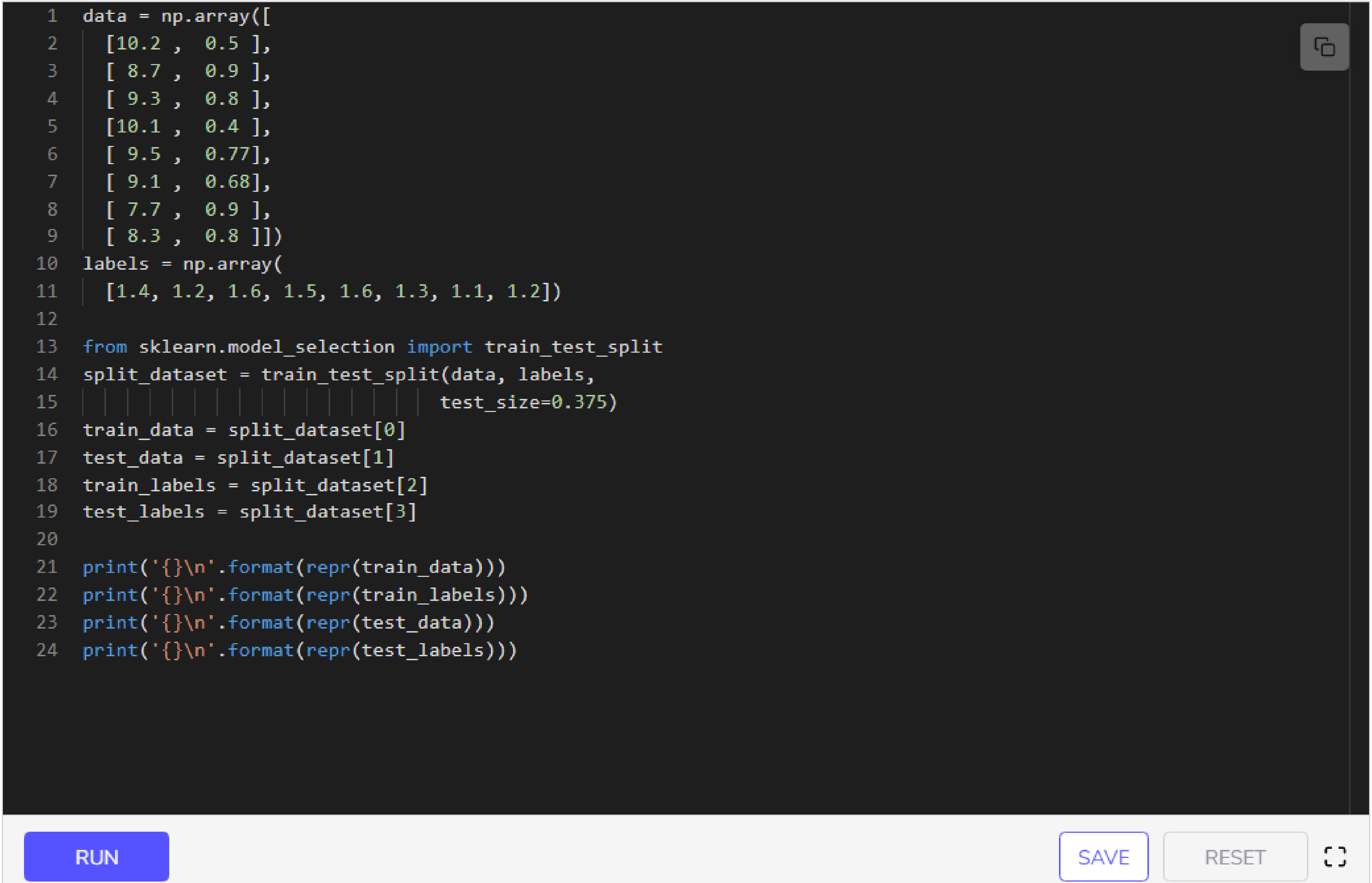
The training set is used for fitting the model on data (i.e. training the model), while the testing set is used for evaluating the model. Therefore, the training set is much larger than the testing set. Exactly how much larger depends on the application and requirements.

Increasing the size of the training set will give more data for the model to be fitted on, which can increase the model's performance. However, because this decreases the size of the testing set, there's a higher chance that the testing set may not be representative of the original dataset (which can lead to inaccurate evaluation).

In general, the testing set is around 10-30% of the original dataset, while the training set makes up the remaining 70-90%.

Note that the `train_test_split` function randomly shuffles the dataset and corresponding labels prior to splitting. This is good practice to remove any systematic orderings in the dataset, which could potentially impact the model into training on the orderings rather than the actual data.

The default size of the testing set is 25% of the original dataset. We can use the `test_size` keyword argument to manually specify the proportion of the original dataset that will go into the testing set.

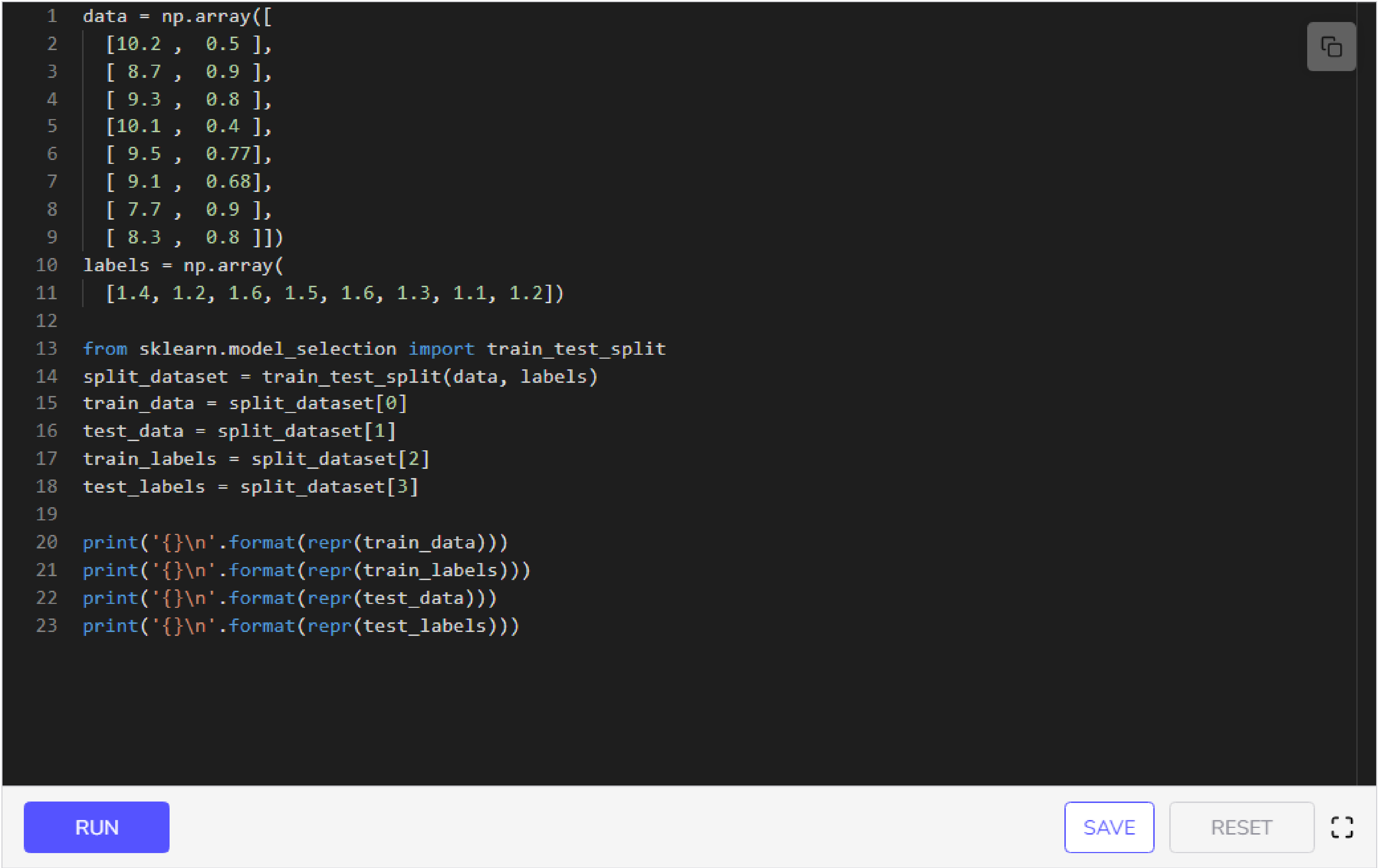


```
data = np.array([
  [10.2 ,  0.5 ],
  [ 8.7 ,  0.9 ],
  [ 9.3 ,  0.8 ],
  [10.1 ,  0.4 ],
  [ 9.5 ,  0.77],
  [ 9.1 ,  0.68],
  [ 7.7 ,  0.9 ],
  [ 8.3 ,  0.8 ]])
labels = np.array(
  [1.4, 1.2, 1.6, 1.5, 1.6, 1.3, 1.1, 1.2])

from sklearn.model_selection import train_test_split
split_dataset = train_test_split(data, labels,
                                test_size=0.375)
train_data = split_dataset[0]
test_data = split_dataset[1]
train_labels = split_dataset[2]
test_labels = split_dataset[3]

print('{}\n'.format(repr(train_data)))
print('{}\n'.format(repr(train_labels)))
print('{}\n'.format(repr(test_data)))
print('{}\n'.format(repr(test_labels)))
```

B. Splitting the dataset

The scikit-learn library provides a nice utility function, called `train_test_split` (which is part of the `model_selection` module) that handles the dataset splitting for us.

The code below demonstrates how to split a dataset into training and testing sets.



```
data = np.array([
  [10.2 ,  0.5 ],
  [ 8.7 ,  0.9 ],
  [ 9.3 ,  0.8 ],
  [10.1 ,  0.4 ],
  [ 9.5 ,  0.77],
  [ 9.1 ,  0.68],
  [ 7.7 ,  0.9 ],
  [ 8.3 ,  0.8 ]])
labels = np.array(
  [1.4, 1.2, 1.6, 1.5, 1.6, 1.3, 1.1, 1.2])

from sklearn.model_selection import train_test_split
split_dataset = train_test_split(data, labels)
train_data = split_dataset[0]
test_data = split_dataset[1]
train_labels = split_dataset[2]
test_labels = split_dataset[3]

print('{}\n'.format(repr(train_data)))
print('{}\n'.format(repr(train_labels)))
print('{}\n'.format(repr(test_data)))
print('{}\n'.format(repr(test_labels)))
```