Digital World (2019)

Week 10, S3: k-NN Classification

Chris Poskitt

SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

neighbours

k-NN Classification

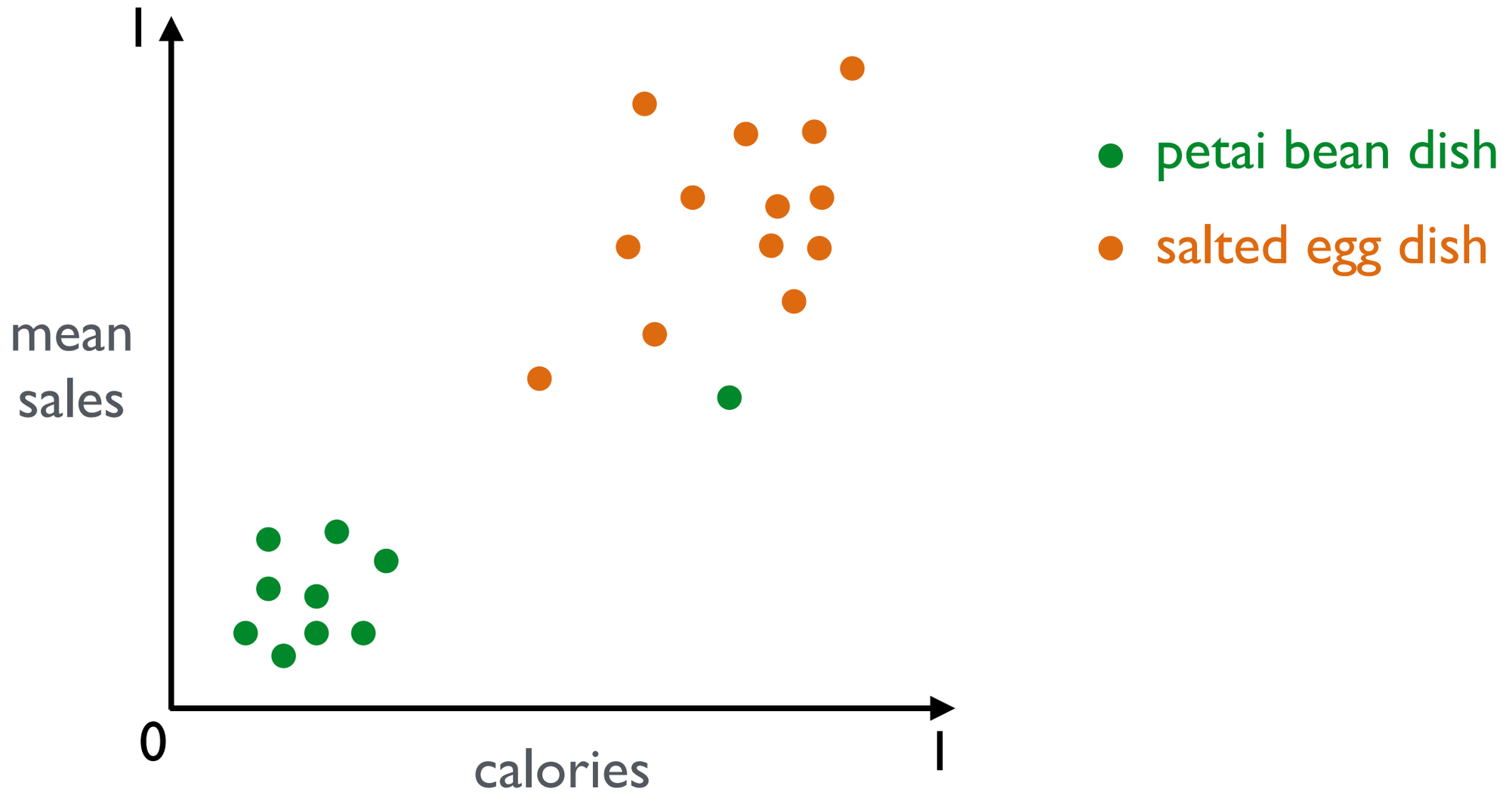
nearest

salted egg dish

petai bean dish

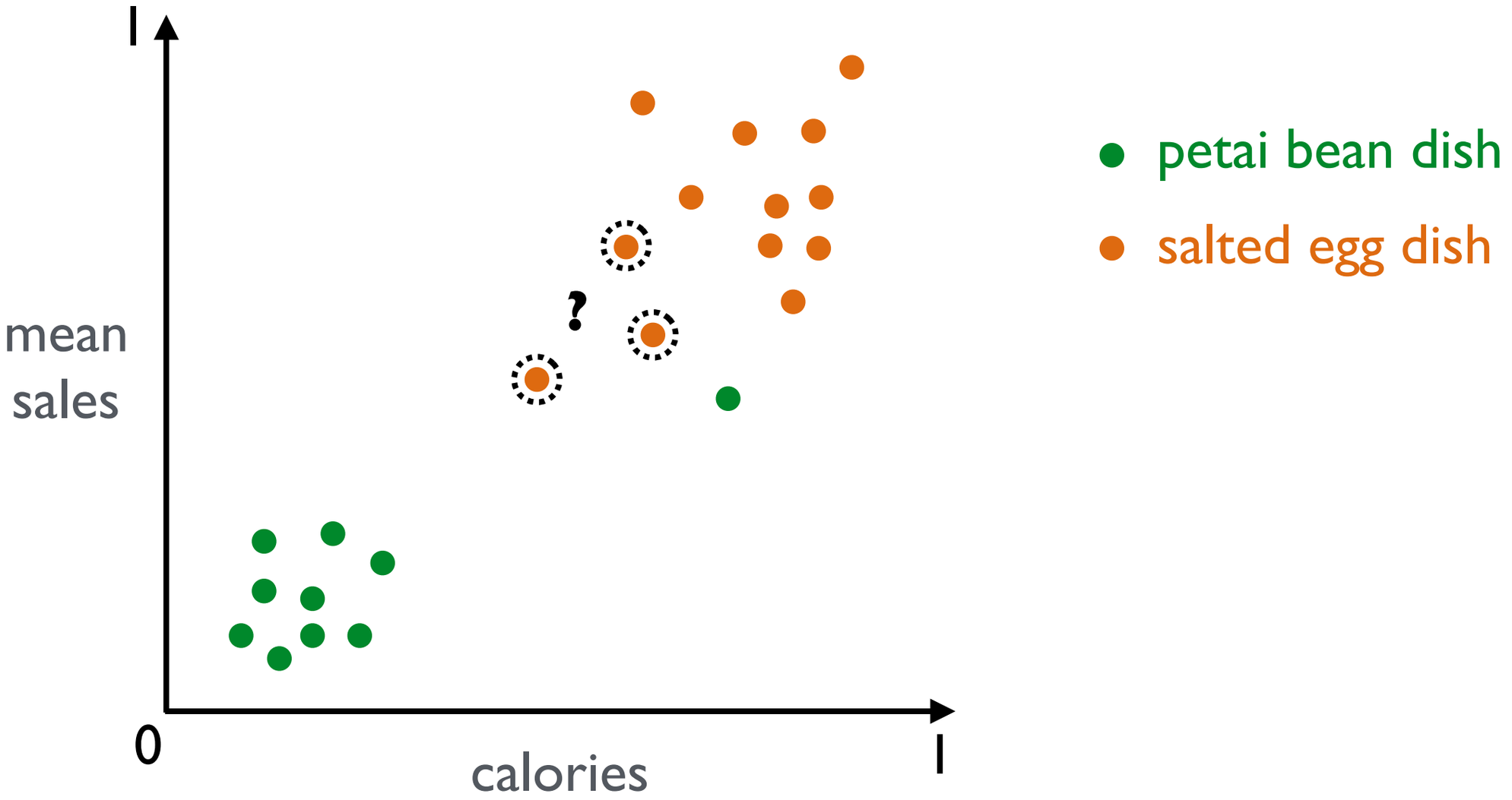
k-NN classification

(with two features)



k-NN classification

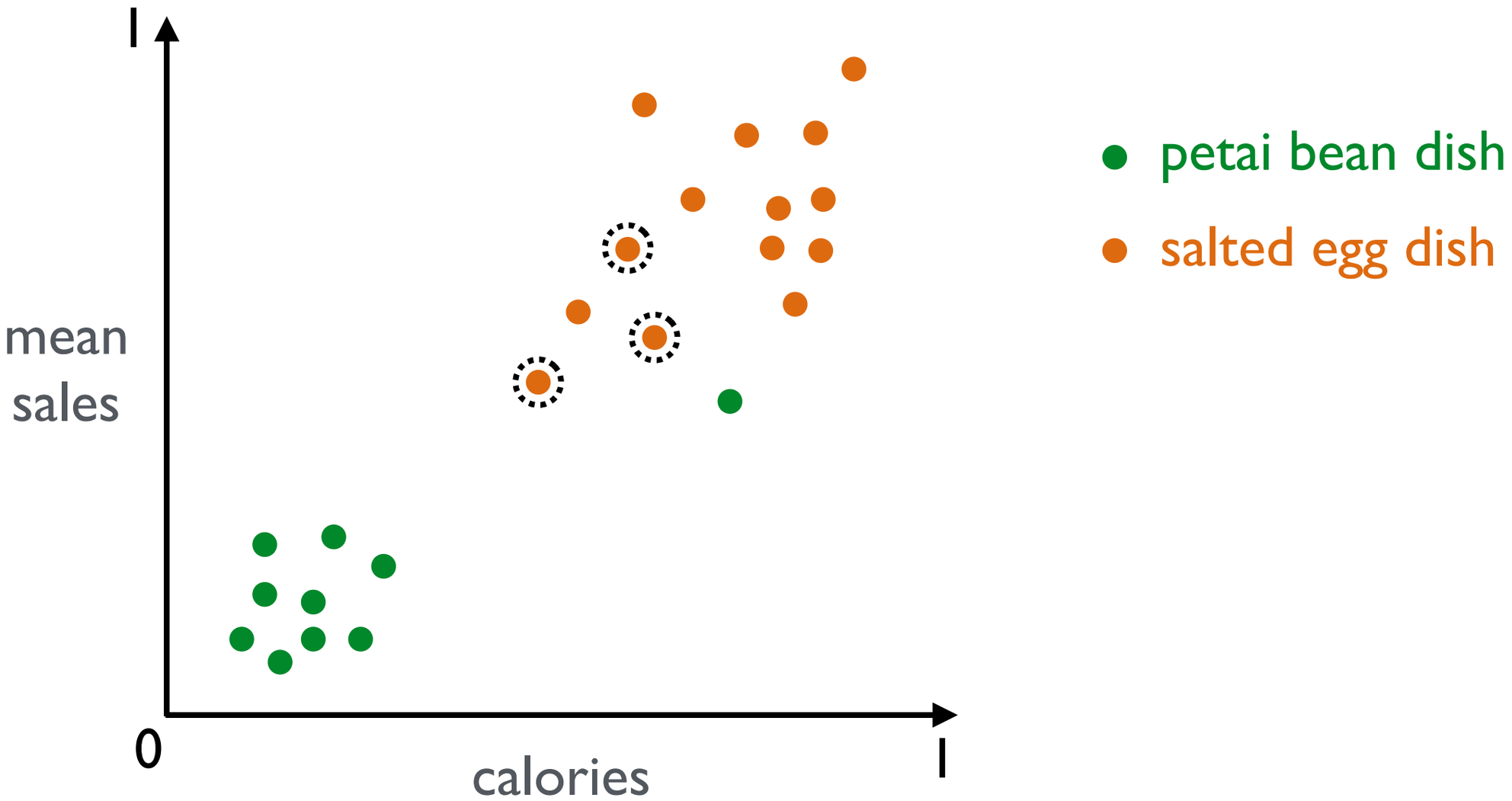
(with two features)



k = 3

k-NN classification

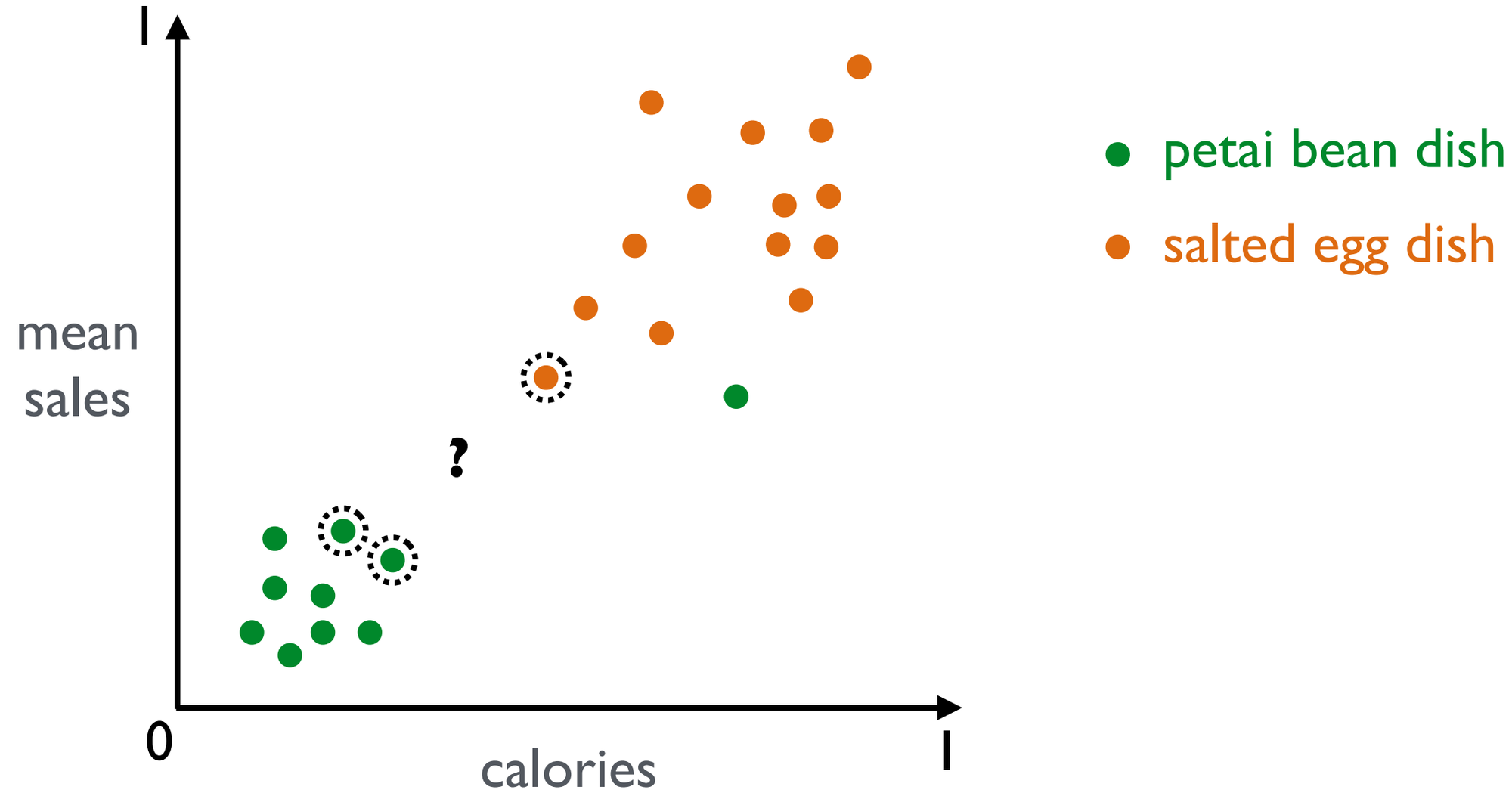
(with two features)



k = 3

k-NN classification

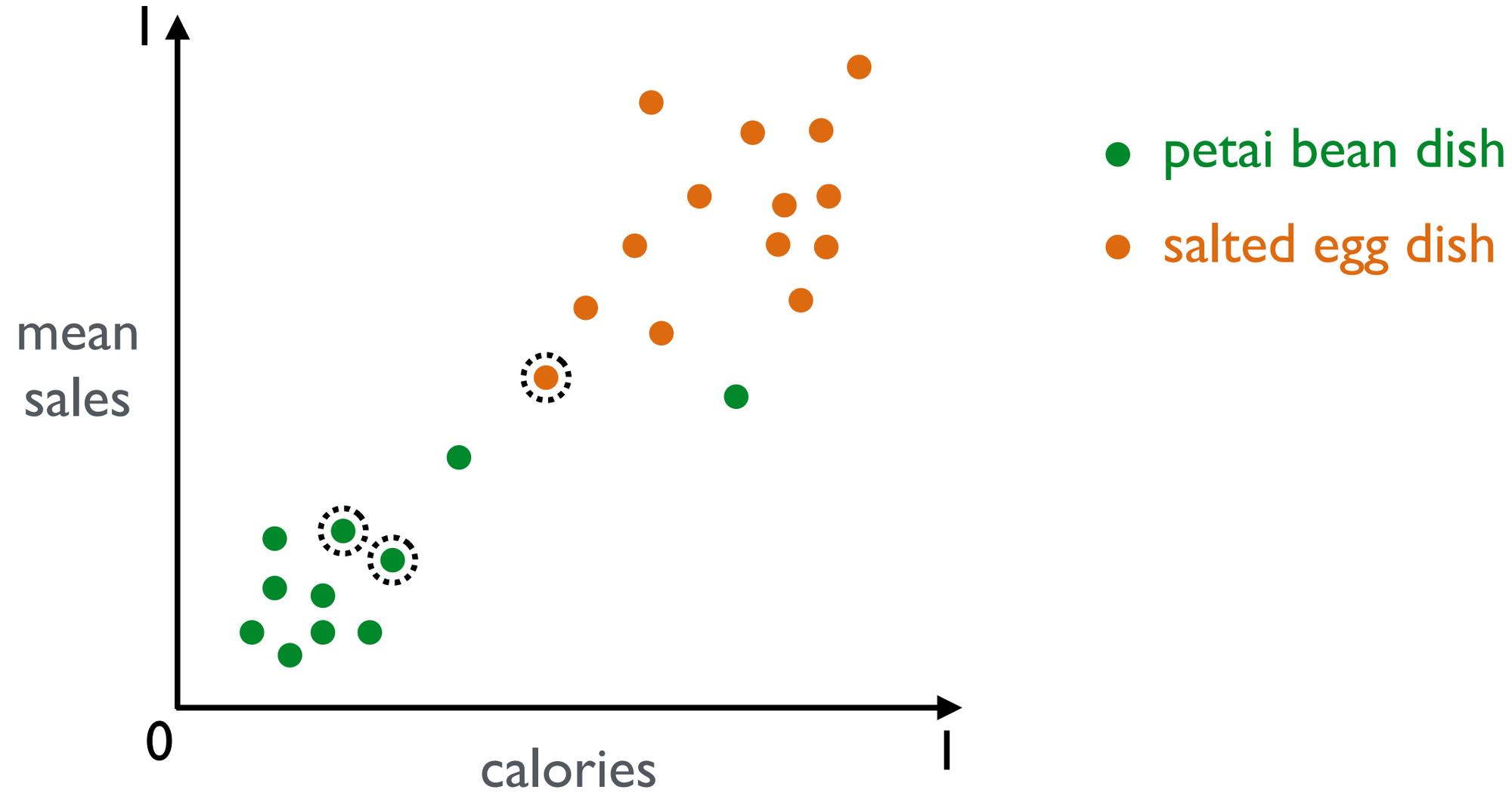
(with two features)



k = 3

k-NN classification

(with two features)



k = 3

Implementing k-NN classification

- we can implement a k-NN classifier using sklearn

  => *in particular, the* ***sklearn.neighbors*** *module*

- normalise the data to values between 0 and 1

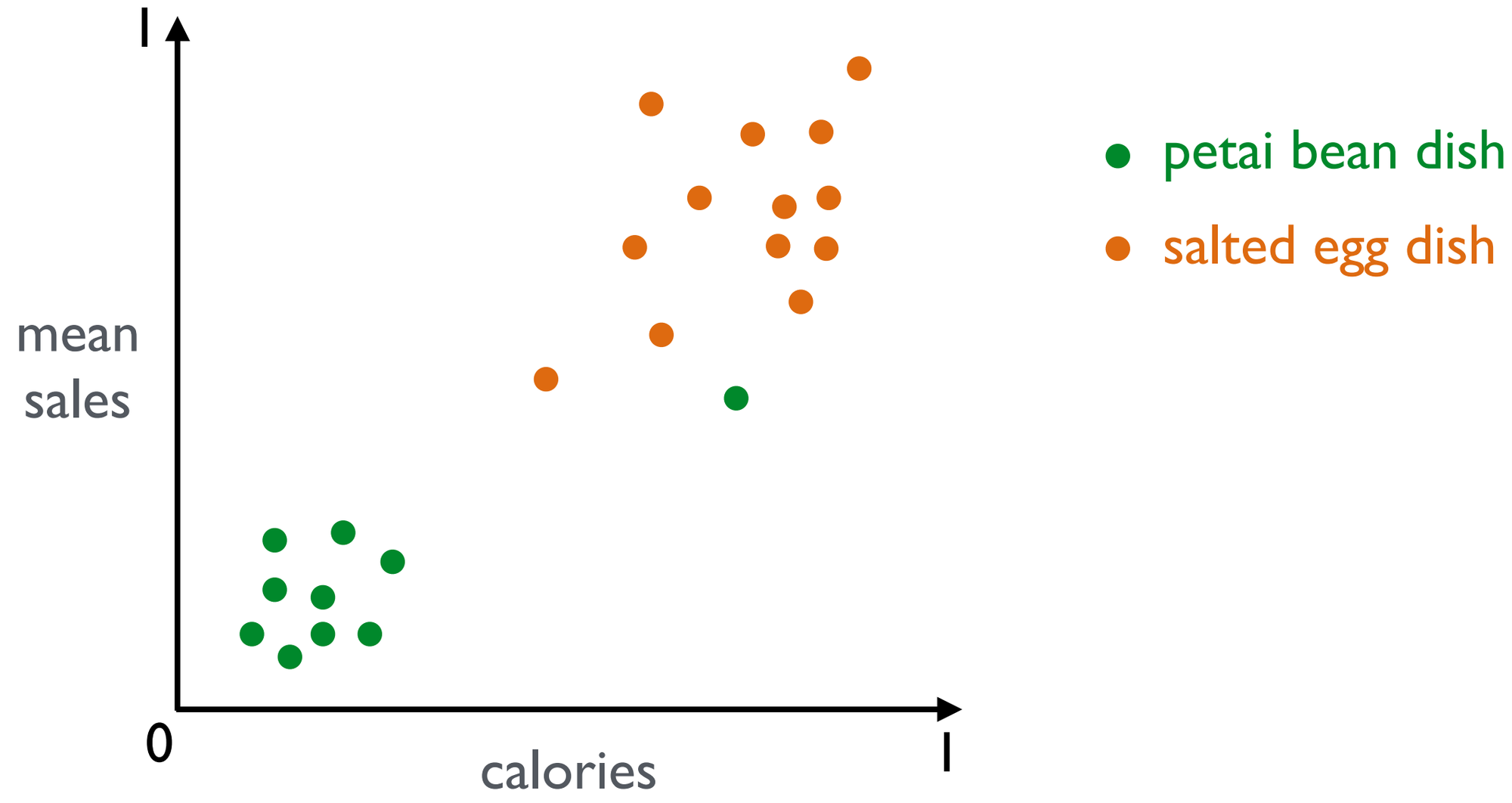
- split your data into training and testing sets

  => *typically a 60% : 40% split*
  => *use the* ***train_test_split()*** *function to do so*

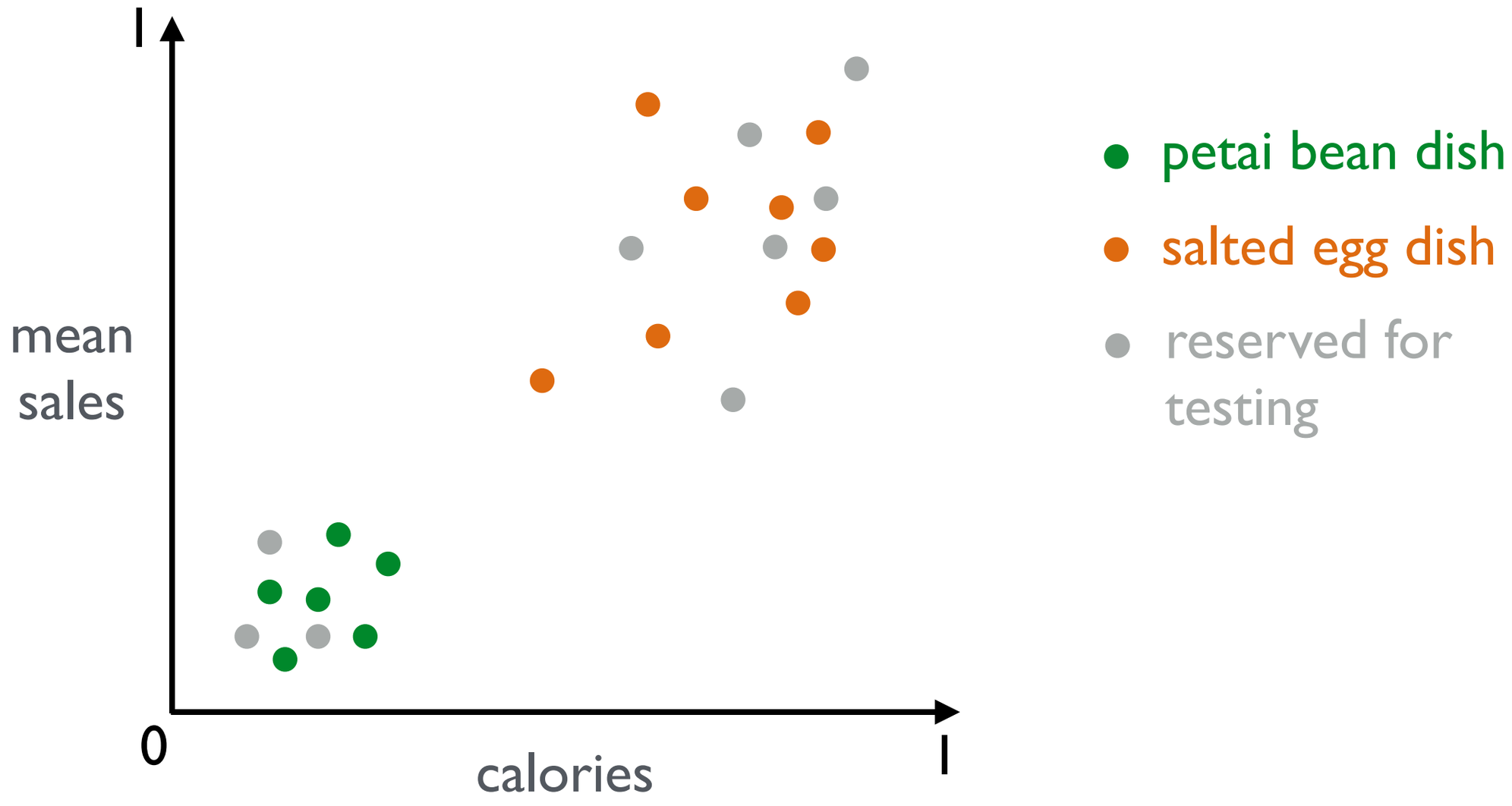
- choose a value of k, then fit the training data

```
clf = neighbors.KNeighborsClassifier(k)
clf.fit(data_train, target_train)
```

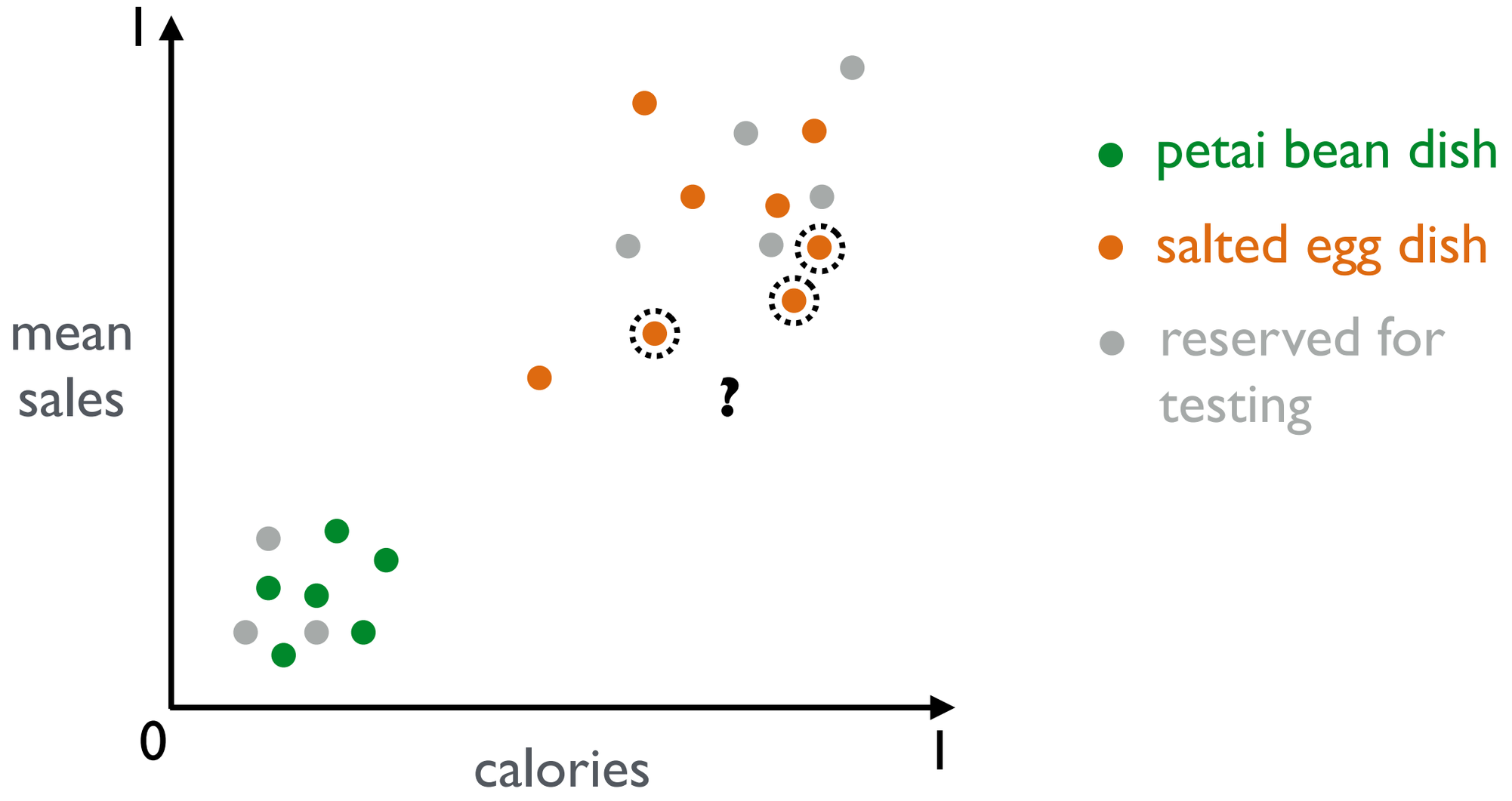
Testing the classifier



Testing the classifier



Applying clf.predict



k = 3

Applying clf.predict

k = 3

k-NN with multiple features

- the breast cancer dataset has two labels: benign, malignant

- unlike our food classifier, it has more than 2 features

  => *radius, texture, perimeter, area, smoothness, …*

- thus the distance measure must generalise to N dimensions

  => *e.g. Euclidean distance*
  => *potential issues for very high dimensions*

- metrics: extract from the confusion matrix

data normalisation

finding the best k

Today: questions CS3, CS4, & CS7 only

k-NN classifier

Summary

- a brief 'black box' introduction to learning with scikit-learn
- *n*-dimensional data represented as NumPy arrays
- with regression we can predict values

  => ***metrics:*** *mean squared error, r2 score, …*

- with classification we can predict labels

  => ***metrics:*** *confusion matrix ⇒ accuracy, sensitivity, false pos. rate*