KNN - Tutorial

Hisham Ihshaish
Bristol, UK
June, 2021

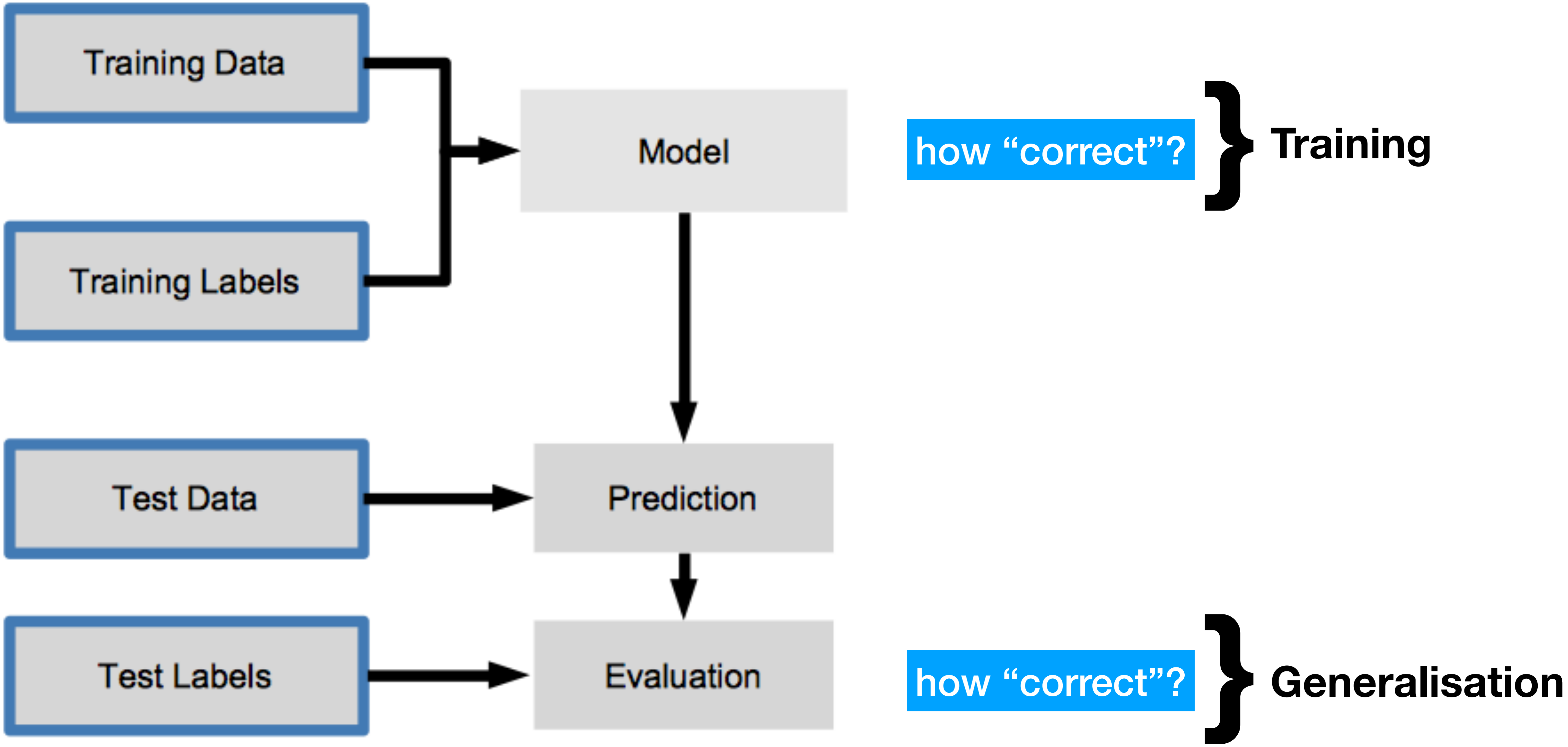
Do we evaluate our ‘model’ manually?

Nah!

lecture 2

Evaluate,

in training and in testing



Data… splitting

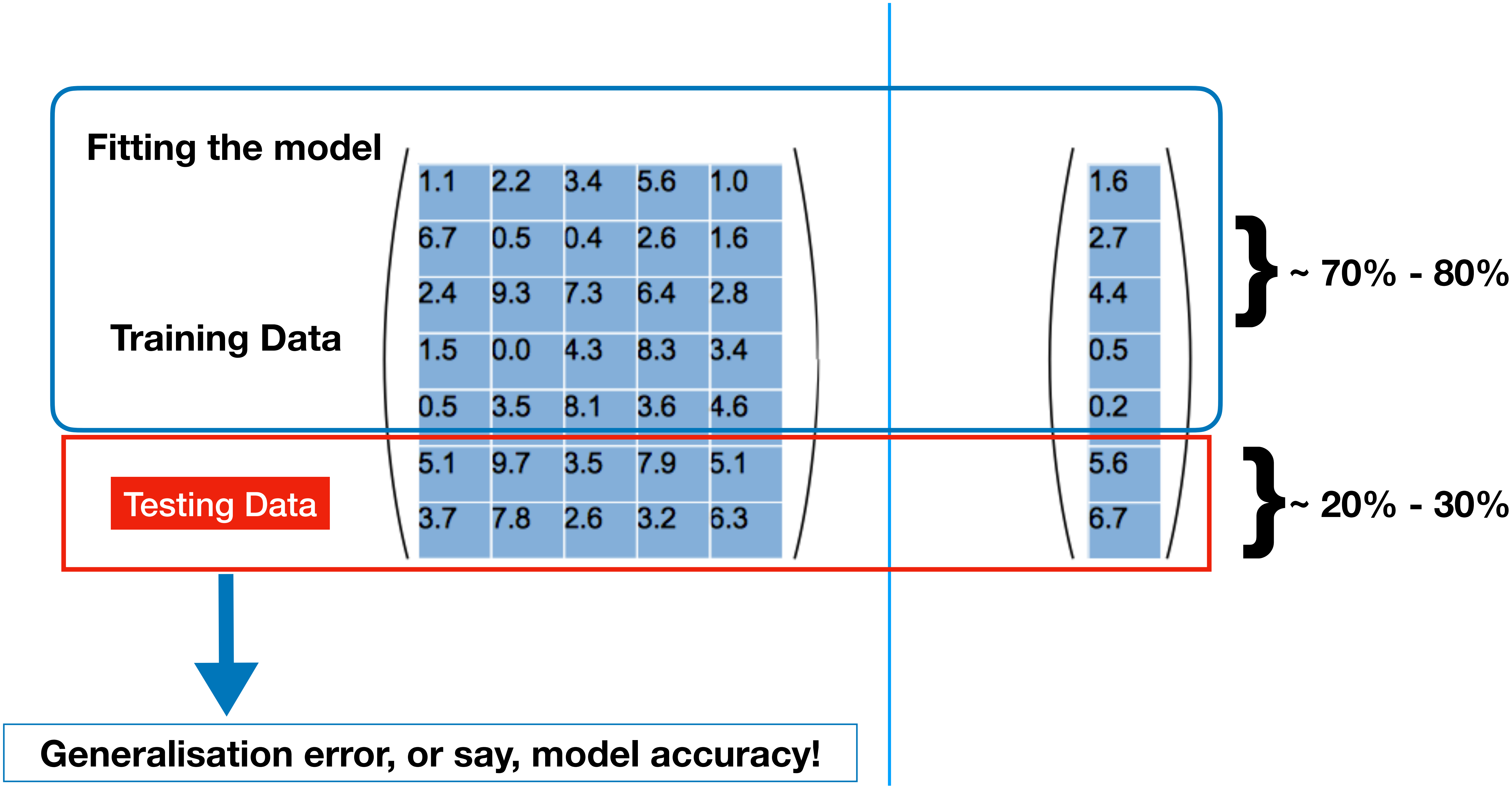
- for **training** and for **testing** (generalisation)

$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix}$$

$$y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

Data… splitting

- for **training** and for **testing** (generalisation)



Fitting the model

Training Data

Testing Data

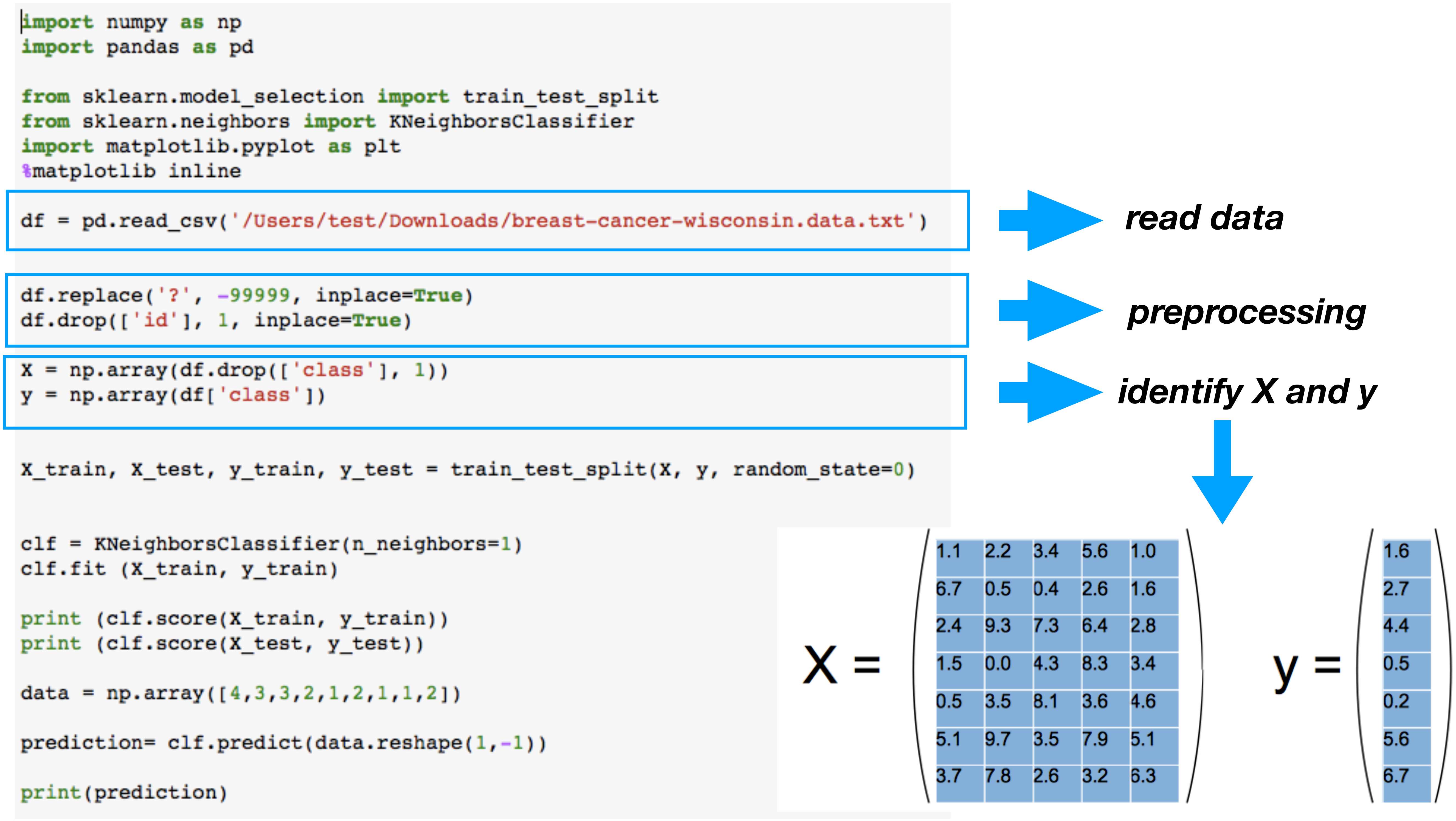
$$\begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix} \qquad \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

~ 70% - 80%

~ 20% - 30%

Generalisation error, or say, model accuracy!

Run example - explained



```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
%matplotlib inline

df = pd.read_csv('/Users/test/Downloads/breast-cancer-wisconsin.data.txt')

df.replace('?', -99999, inplace=True)
df.drop(['id'], 1, inplace=True)

X = np.array(df.drop(['class'], 1))
y = np.array(df['class'])


X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)


clf = KNeighborsClassifier(n_neighbors=1)
clf.fit (X_train, y_train)

print (clf.score(X_train, y_train))
print (clf.score(X_test, y_test))

data = np.array([4,3,3,2,1,2,1,1,2])

prediction= clf.predict(data.reshape(1,-1))

print(prediction)
```

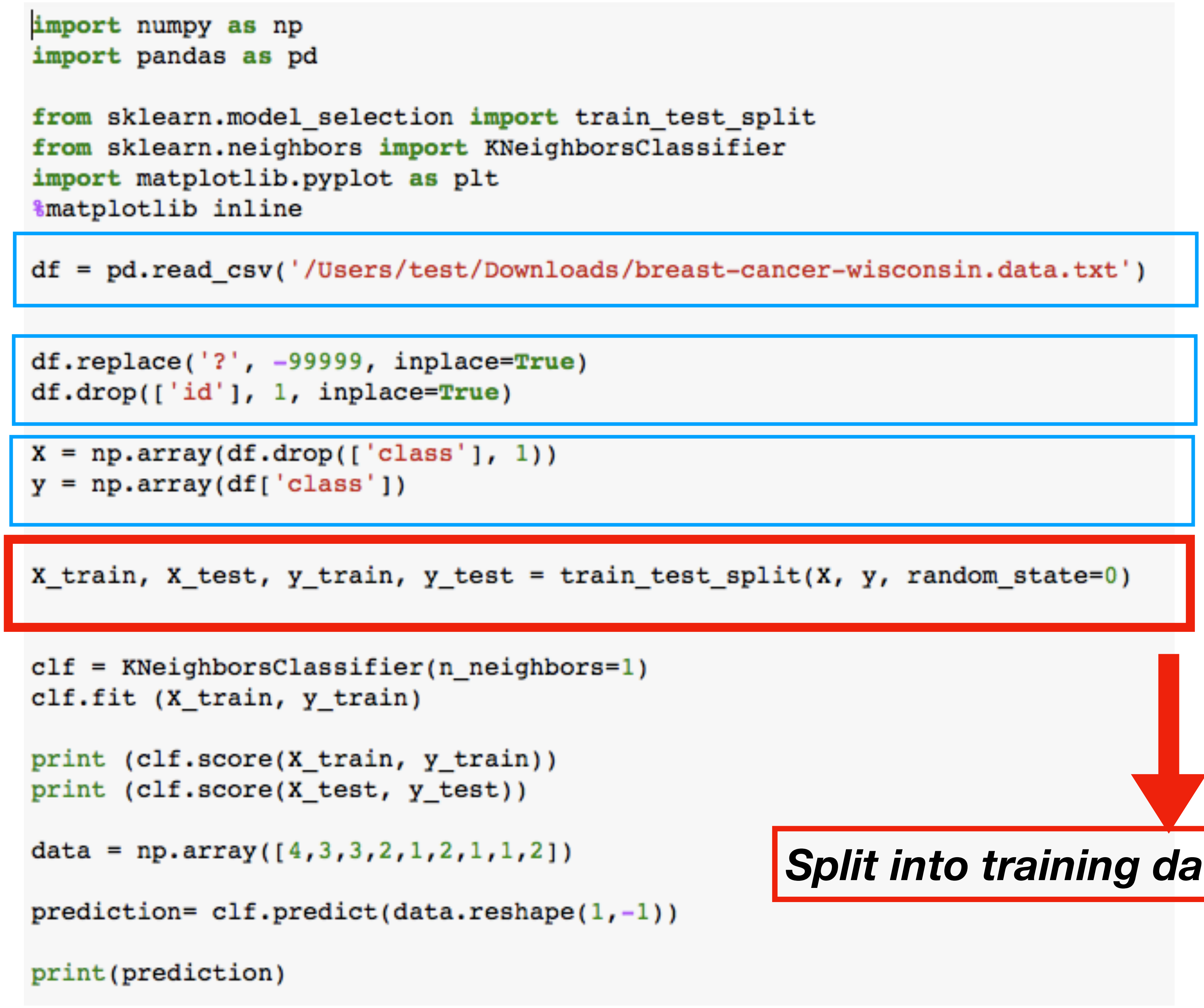
read data

preprocessing

identify X and y

$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix} \qquad y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

Run example - explained



```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
%matplotlib inline

df = pd.read_csv('/Users/test/Downloads/breast-cancer-wisconsin.data.txt')

df.replace('?', -99999, inplace=True)
df.drop(['id'], 1, inplace=True)

X = np.array(df.drop(['class'], 1))
y = np.array(df['class'])

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

clf = KNeighborsClassifier(n_neighbors=1)
clf.fit (X_train, y_train)

print (clf.score(X_train, y_train))
print (clf.score(X_test, y_test))

data = np.array([4,3,3,2,1,2,1,1,2])

prediction= clf.predict(data.reshape(1,-1))

print(prediction)
```

read data

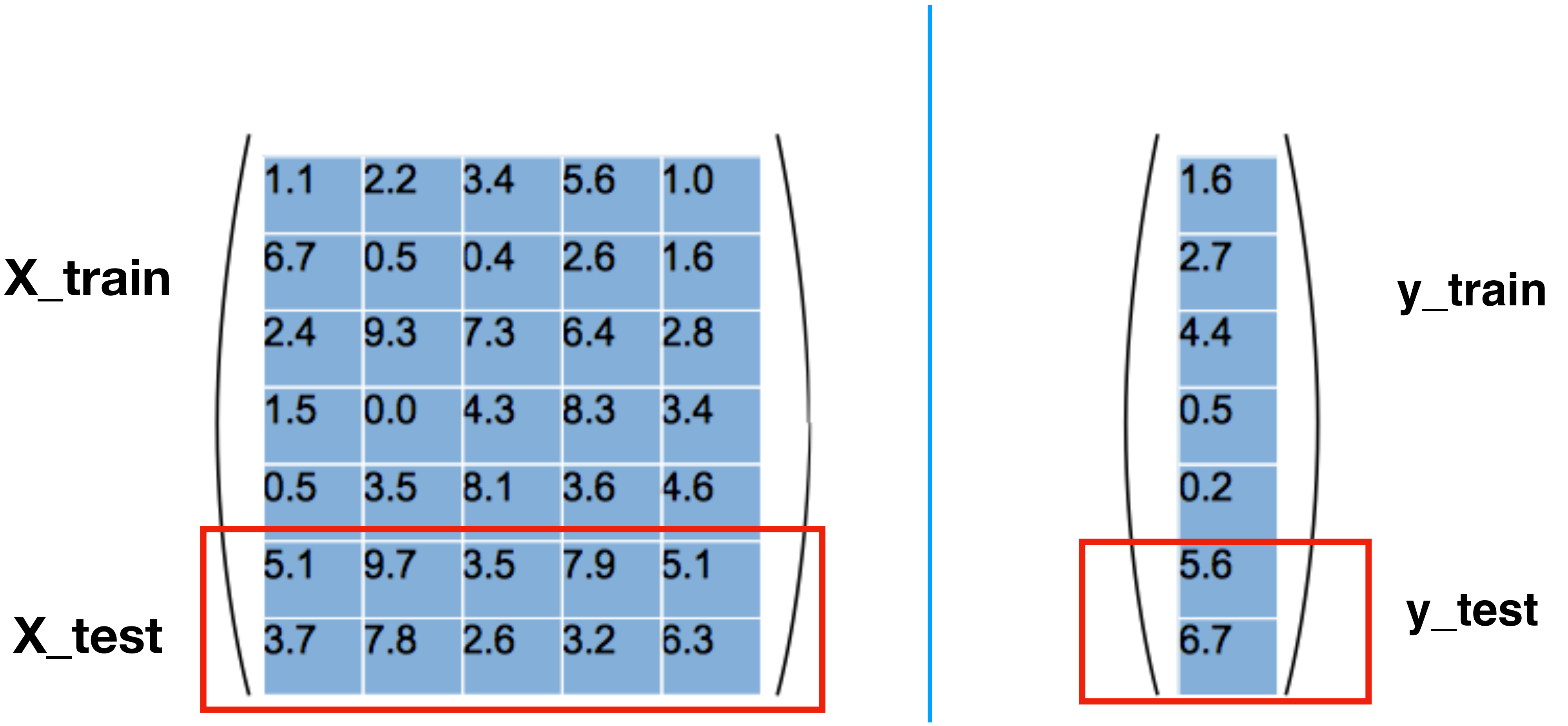
preprocessing

$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix} \qquad y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

Split into training data and testing data

Data… splitting

- for **training** and for **testing** (generalisation)



X_train

1.1	2.2	3.4	5.6	1.0
6.7	0.5	0.4	2.6	1.6
2.4	9.3	7.3	6.4	2.8
1.5	0.0	4.3	8.3	3.4
0.5	3.5	8.1	3.6	4.6

X_test

5.1	9.7	3.5	7.9	5.1
3.7	7.8	2.6	3.2	6.3

y_train

1.6
2.7
4.4
0.5
0.2

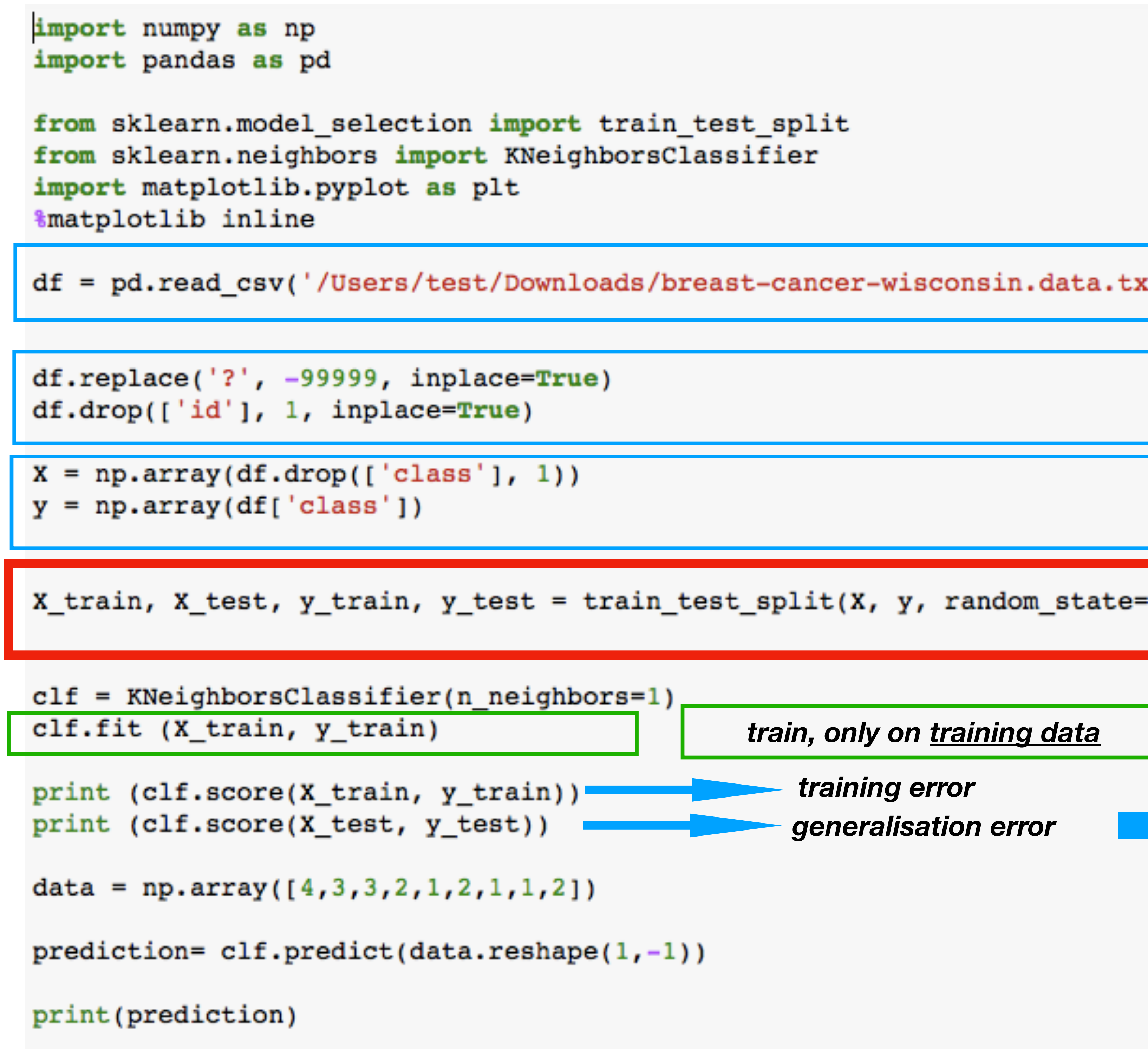
y_test

5.6
6.7

sklearn >>

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

here goes what we did last week



```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
%matplotlib inline

df = pd.read_csv('/Users/test/Downloads/breast-cancer-wisconsin.data.txt')

df.replace('?', -99999, inplace=True)
df.drop(['id'], 1, inplace=True)

X = np.array(df.drop(['class'], 1))
y = np.array(df['class'])

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

clf = KNeighborsClassifier(n_neighbors=1)
clf.fit (X_train, y_train)

print (clf.score(X_train, y_train))
print (clf.score(X_test, y_test))

data = np.array([4,3,3,2,1,2,1,1,2])

prediction= clf.predict(data.reshape(1,-1))

print(prediction)
```

read data

preprocessing

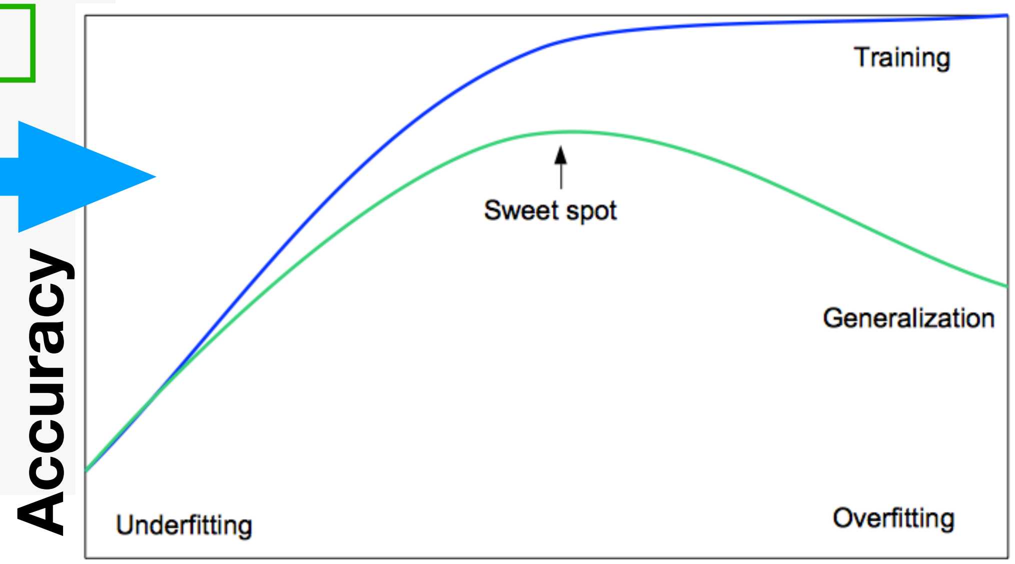
$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix} \quad y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

Split into training data and testing data

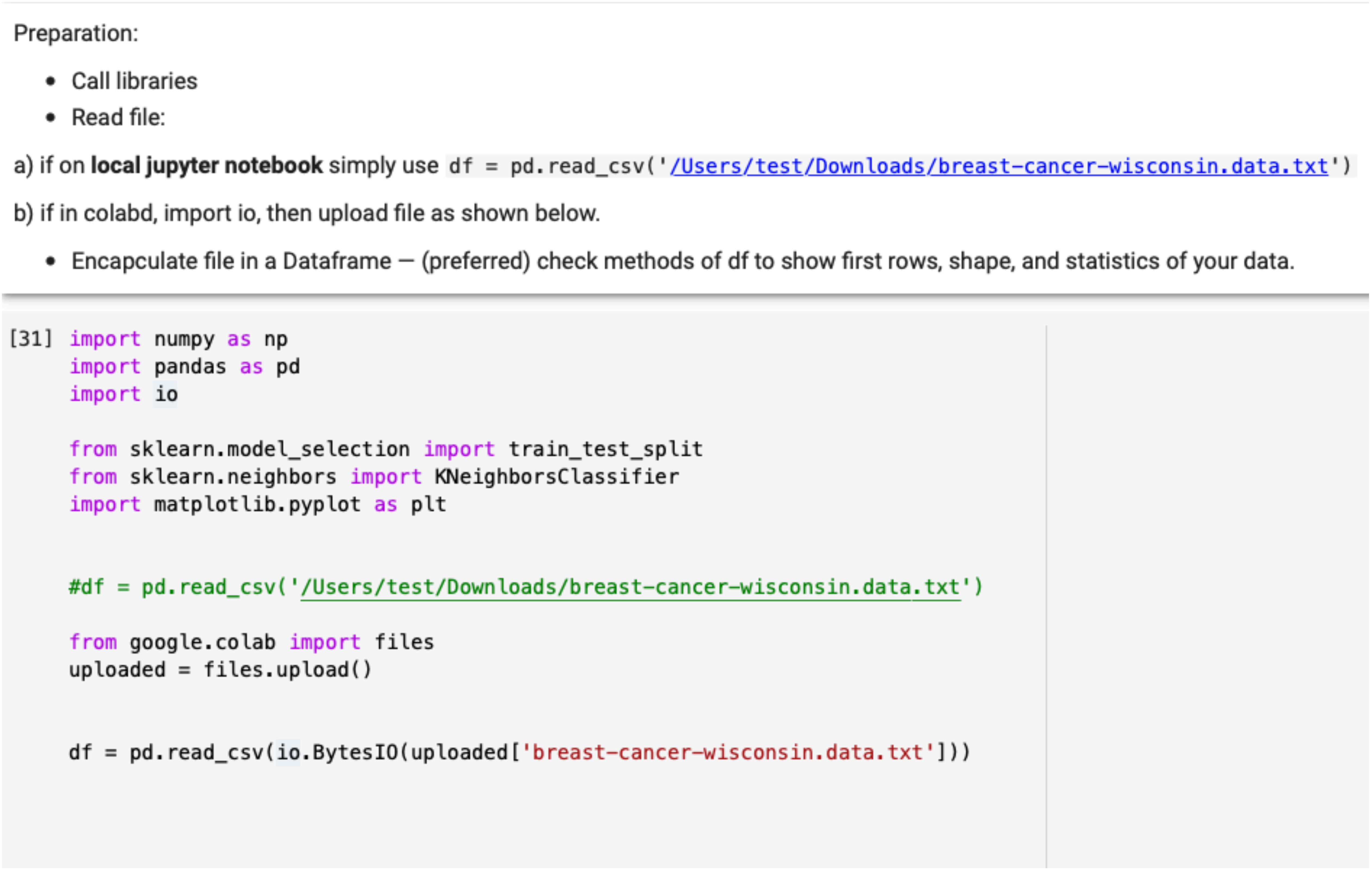
*train, only on **training data***

training error

generalisation error



note on file upload



Preparation:

- Call libraries
- Read file:

a) if on **local jupyter notebook** simply use `df = pd.read_csv('/Users/test/Downloads/breast-cancer-wisconsin.data.txt')`

b) if in colabd, import io, then upload file as shown below.

- Encapculate file in a Dataframe — (preferred) check methods of df to show first rows, shape, and statistics of your data.

[31]

```
import numpy as np
import pandas as pd
import io

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt


#df = pd.read_csv('/Users/test/Downloads/breast-cancer-wisconsin.data.txt')

from google.colab import files
uploaded = files.upload()


df = pd.read_csv(io.BytesIO(uploaded['breast-cancer-wisconsin.data.txt']))
```