Loading the data

In [ ]:

```
# import load_iris function from datasets module
import numpy as np
from sklearn.datasets import load_iris

# save "bunch" object containing iris dataset and its attributes
iris = load_iris()

# store feature matrix in "X"
X = iris.data

# store response vector in "y"
y = iris.target
```

In [ ]:

```
# print the shapes of X and y
print X.shape
print y.shape
```

scikit-learn 4-step modeling pattern

Step 1: Import the class you plan to use

In [ ]:

```
from sklearn.neighbors import KNeighborsClassifier
```

Step 2: "Instantiate" the "estimator"

- "Estimator" is scikit-learn's term for model
- "Instantiate" means "make an instance of"

In [ ]:

```
knn = KNeighborsClassifier(n_neighbors=1)
```

- Name of the object does not matter
- Can specify tuning parameters (aka "hyperparameters") during this step
- All parameters not specified are set to their defaults

In [ ]:

```
print knn
```

Step 3: Fit the model with data (aka "model training")

- Model is learning the relationship between X and y
- Occurs in-place

In [ ]:

```
knn.fit(X, y)
```

Step 4: Predict the response for a new observation

- New observations are called "out-of-sample" data
- Uses the information it learned during the model training process

In [ ]:

```
X_new = np.array([3, 5, 4, 2]).reshape(1,4)
knn.predict(X_new)
```

- Returns a NumPy array
- Can predict for multiple observations at once

In [ ]:

```
X_new =np.array([[3, 5, 4, 2], [5, 4, 3, 2]])
knn.predict(X_new)
```

Using a different value for K

In [ ]:

```
# instantiate the model (using the value K=5)
knn = KNeighborsClassifier(n_neighbors=5)

# fit the model with data
knn.fit(X, y)

# predict the response for new observations
knn.predict(X_new)
```

Using a different classification model

In [ ]:

```
# import the class
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression()

# fit the model with data
logreg.fit(X, y)

# predict the response for new observations
logreg.predict(X_new)
```