Exercise: Supervised and Unsupervised Machine Learning

Using the iris dataset from the previous lesson, we're going to create two models, one supervised, one unsupervised, and compare how their predictions differ.

Complete the notebook by filling in the code where there are `?`.

In [2]:

```
import numpy as np
import pandas as pd
import sklearn
from sklearn import datasets
```

In [3]:

```
# Load in the iris dataset
iris = datasets.load_iris()
```

In [8]:

```
# Create the iris `data` dataset as a dataframe and name the columns with `feature_names`
df = pd.DataFrame(iris['data'], columns=iris['feature_names'])

# Include the target as well
df['target'] = iris['target']
```

In [9]:

```
# Check your dataframe by `.head()`
df.head()
```

Out[9]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [16]:

```
# Target values as an array to compare against supervised and unsupervised
target = np.array(df[,df["target"]])
target
```

Out[16]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

Supervised ML

In [17]:

```
from sklearn.linear_model import LinearRegression
```

In [20]:

```
# initialize and fit a linear regression model
reg = LinearRegression().fit(df[iris['feature_names']],target)
```

In [21]:

```
# Scoring of the linear regression model, but slighly deceiving since the iris dataset is classifying not regression
reg.score(df[iris['feature_names']], target)
```

Out[21]:

```
0.93042236753315966
```

In [22]:

```
# regression output floating point numbers
reg.predict(df[iris['feature_names']])
```

Out[22]:

```
array([ -8.26582725e-02,  -3.85897565e-02,  -4.81896914e-02,
         1.26087761e-02,  -7.61081708e-02,   5.68023484e-02,
         3.76259158e-02,  -4.45599433e-02,   2.07050198e-02,
        -8.13030749e-02,  -1.01728663e-01,   8.84875996e-05,
        -8.86050221e-02,  -1.01834705e-01,  -2.26997797e-01,
        -4.36405904e-02,  -3.39982044e-02,  -2.16688605e-02,
        -3.26854579e-02,  -1.22408563e-02,  -4.30562522e-02,
         5.31726003e-02,  -1.23012138e-01,   1.77258467e-01,
         6.81889023e-02,  -4.16362637e-03,   1.00119019e-01,
        -7.09322806e-02,  -8.92083742e-02,   1.99107233e-02,
         1.33606216e-02,   3.35222953e-02,  -1.58465961e-01,
        -1.57523171e-01,  -8.13030749e-02,  -1.03812269e-01,
        -1.49254996e-01,  -8.13030749e-02,  -6.41916305e-03,
        -5.55340896e-02,  -3.33948524e-02,   7.45644153e-02,
        -1.52672524e-02,   2.17673798e-01,   1.39549109e-01,
         3.33738018e-02,  -5.05301301e-02,  -1.45154068e-02,
        -9.07545163e-02,  -6.28360368e-02,   1.20308259e+00,
         1.28451660e+00,   1.32487047e+00,   1.18762080e+00,
         1.31393877e+00,   1.25705298e+00,   1.39745639e+00,
         9.07172433e-01,   1.17656176e+00,   1.24113634e+00,
         9.59294742e-01,   1.28013501e+00,   9.54205881e-01,
         1.31512204e+00,   1.05930184e+00,   1.17232866e+00,
         1.38115786e+00,   9.76734088e-01,   1.35070534e+00,
         1.02311961e+00,   1.59045598e+00,   1.09965570e+00,
         1.41725961e+00,   1.19756726e+00,   1.13040963e+00,
         1.18772685e+00,   1.26542720e+00,   1.49592176e+00,
         1.34168532e+00,   8.55931450e-01,   1.01581766e+00,
         9.32128108e-01,   1.05331264e+00,   1.54772365e+00,
         1.40310615e+00,   1.38055451e+00,   1.30141848e+00,
         1.19062819e+00,   1.16837848e+00,   1.17877271e+00,
         1.20415981e+00,   1.28799785e+00,   1.08043682e+00,
         9.00622332e-01,   1.20435076e+00,   1.11911506e+00,
         1.18452852e+00,   1.15235793e+00,   8.73689093e-01,
         1.16625243e+00,   2.24146289e+00,   1.75264018e+00,
         1.90028407e+00,   1.74143264e+00,   2.00441822e+00,
         2.00431431e+00,   1.60207593e+00,   1.79059214e+00,
         1.76063251e+00,   2.15212358e+00,   1.71469034e+00,
         1.73219558e+00,   1.84240596e+00,   1.81075169e+00,
         2.05316319e+00,   1.95403300e+00,   1.69236016e+00,
         2.04163735e+00,   2.20111558e+00,   1.48615432e+00,
         1.98996282e+00,   1.78575356e+00,   1.96389898e+00,
```

```
       1.59137976e+00,   1.88550825e+00,   1.72019374e+00,
       1.57522972e+00,   1.60005592e+00,   1.91785077e+00,
       1.56166273e+00,   1.79963117e+00,   1.82960982e+00,
       1.97884018e+00,   1.44938775e+00,   1.53269542e+00,
       2.00181829e+00,   2.08524888e+00,   1.69891026e+00,
       1.58832992e+00,   1.80430763e+00,   2.05462443e+00,
       1.85818604e+00,   1.75264018e+00,   2.04633725e+00,
       2.12946589e+00,   1.90725851e+00,   1.68391740e+00,
       1.74623857e+00,   1.98983334e+00,   1.66740449e+00])
```

Unsupervised ML

In [23]:

```
from sklearn.cluster import KMeans
```

In [24]:

```
# We already know the number of clusters, we can use during fit, hint: it's the number of classes
kmeans = KMeans(n_clusters=3, random_state=0).fit(df[iris["feature_names"]])
```

In [25]:

```
# Print the labels to see what value is in what cluster
kmeans.labels_
```

Out[25]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0, 0,
       2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2,
       0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0], dtype=int32)
```

In [28]:

```
# What happens if we cluster more than the actual classes?
kmeans = KMeans(n_clusters=4, random_state=0).fit(df[iris["feature_names"]])
```

Out[28]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=4, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=0, tol=0.0001, verbose=0)
```

In [27]:

```
# Print the labels to see what value is in what cluster
kmeans.labels_
```

Out[27]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 3, 3, 3, 0, 3, 0, 3, 0, 3, 0, 0, 0, 0, 3, 0, 3, 0, 0, 3,
       0, 3, 0, 3, 3, 3, 3, 3, 3, 3, 0, 0, 0, 0, 3, 0, 3, 3, 3, 0, 0, 0, 3,
       0, 0, 0, 0, 0, 3, 0, 0, 2, 3, 2, 2, 2, 2, 0, 2, 2, 2, 3, 3, 2, 3, 3,
       2, 2, 2, 2, 3, 2, 3, 2, 3, 2, 2, 3, 3, 2, 2, 2, 2, 2, 3, 3, 2, 2, 2,
       3, 2, 2, 2, 3, 2, 2, 2, 3, 3, 2, 3], dtype=int32)
```