Exercise 2: Clustering 2D points

From the course *Transition to Data Science*. Buy the entire course for just $10 for many more exercises and helpful video lectures.

From the scatter plot of the previous exercise, you saw that the points seem to separate into 3 clusters. Now create a KMeans model to find 3 clusters, and fit it to the data points from the previous exercise. After the model has been fit, obtain the cluster labels for points, and also for some new points using the `.predict()` method.

You are given the array `points` from the previous exercise, and also an array `new_points`.

Step 1: Load the dataset *(written for you)*.

```
In [1]: import pandas as pd

        df = pd.read_csv('../datasets/ch1ex1.csv')
        points = df.values

        new_df = pd.read_csv('../datasets/ch1ex2.csv')
        new_points = new_df.values
```

Step 2: Import `KMeans` from `sklearn.cluster`

```
In [2]: from sklearn.cluster import KMeans
```

Step 3: Using `KMeans()`, create a `KMeans` instance called `model` to find `3` clusters. To specify the number of clusters, use the `n_clusters` keyword argument

```
In [3]: model = KMeans(n_clusters=3)
```

Step 4: Use the `.fit()` method of `model` to fit the model to the array of points `points`.

```
In [4]: model.fit(points)

Out[4]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
            n_clusters=3, n_init=10, n_jobs=1, precompute_distances='auto',
            random_state=None, tol=0.0001, verbose=0)
```

Step 5: Use the `.predict()` method of `model` to predict the cluster labels of `points`, assigning the result to `labels`.

```
In [5]: labels = model.predict(points)
```

Step 6: Print out the labels, and have a look at them! *(In the next exercise, I'll show you how to visualise this clustering better.)*

© 2016 Benjamin Wilson

```
In [6]: print(labels)
```

```
[0 1 2 2 1 1 2 0 1 1 2 0 1 2 1 0 2 2 0 2 1 0 1 0 0 1 0 0 0 1 2 2 2 1 0 1 0
 0 1 0 0 2 1 1 1 0 0 2 0 2 2 2 0 0 0 1 0 0 1 2 1 0 0 2 2 1 2 1 1 0 2 1 2 0
 2 1 0 0 0 2 0 1 2 1 1 1 1 0 0 2 1 2 1 0 0 0 2 1 1 2 1 0 1 2 0 2 2 2 1 1 0
 1 2 1 1 1 0 1 2 2 0 0 0 0 0 1 2 0 1 1 2 2 1 0 1 0 2 1 2 0 2 2 0 2 2 0 2 1
 0 0 0 2 2 1 2 1 0 0 2 1 2 2 2 1 0 0 1 2 2 0 0 2 0 0 1 0 2 2 2 0 0 2 0 2 2
 0 1 2 0 0 0 0 1 2 0 1 1 1 0 1 0 0 1 2 2 0 2 0 0 1 1 0 2 1 2 0 2 1 0 1 1 1
 1 2 2 2 0 0 1 0 2 1 0 0 1 0 2 2 2 2 2 1 0 0 2 2 0 1 2 1 1 0 0 1 1 1 0 2 0
 1 0 2 2 2 2 2 0 0 1 0 0 1 2 2 1 0 2 2 1 1 0 0 0 1 1 0 2 1 1 2 0 0 0 1 0 0
 0 1 1 1]
```

Step 7: Use the `.predict()` method of `model` to predict the cluster labels of `new_points`, assigning the result to `new_labels`. Notice that KMeans can assign previously unseen points to the clusters it has already found!

```
In [7]: new_labels = model.predict(new_points)
```

© 2016 Benjamin Wilson