A Simple Example of Clustering

You are given much more country data. Using the same methodology as the one in the lecture, group all the countries in 2 clusters.

Try with other numbers of clusters and see if they match your expectations. Maybe 7 is going to be a cool one!

Plot the data using the *c* parameter to separate the data by the clusters we defined.

Note: c stands for color

Import the relevant libraries

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
sns.set()
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

Load the data

Load data from the csv file: 'Countries.csv'.

In [9]:
```
data = pd.read_csv('Countries-exercise.csv')
data.head()
```

Out[9]:

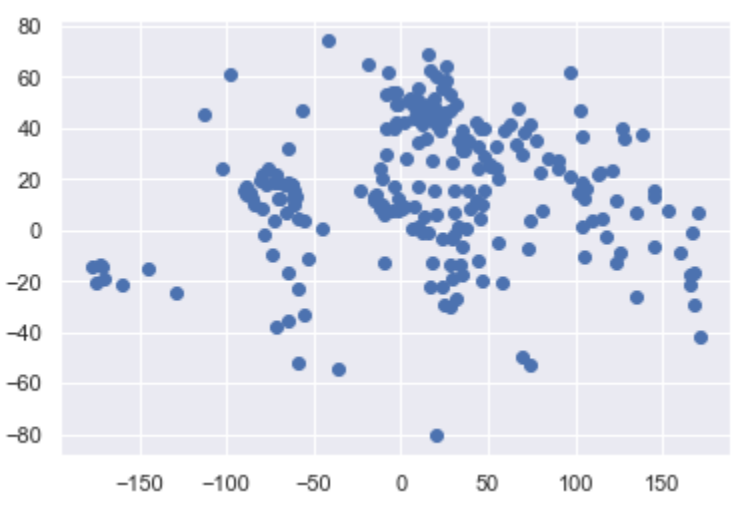
	name	Longitude	Latitude
0	Aruba	-69.982677	12.520880
1	Afghanistan	66.004734	33.835231
2	Angola	17.537368	-12.293361
3	Anguilla	-63.064989	18.223959
4	Albania	20.049834	41.142450

Plot the data

Plot the 'Longtitude' and 'Latitude' columns.

In [7]:
```
plt.scatter(data['Longitude'], data['Latitude'])
```

Out[7]:
```
<matplotlib.collections.PathCollection at 0x217ac4182b0>
```



Select the features

Create a copy of that data and remove all parameters apart from Longitude and Latitude.

In [11]:
```
x = data.iloc[:,1:]
x.head()
```

Out[11]:

	Longitude	Latitude
0	-69.982677	12.520880
1	66.004734	33.835231
2	17.537368	-12.293361
3	-63.064989	18.223959
4	20.049834	41.142450

Clustering

Assume there are only two clusters.

In [97]:
```
kmeans = KMeans(3)
```

In [31]:
```
kmeans.fit(x)
```

Out[31]:
```
KMeans(n_clusters=3)
```

Clustering Resutls

In [32]:
```
identified_clusters = kmeans.fit_predict(x)
identified_clusters
```

Out[32]:
```
array([0, 2, 2, 0, 2, 2, 2, 2, 0, 2, 0, 2, 1, 1, 0, 1, 2, 2, 2, 2, 2, 2,
       1, 2, 2, 0, 0, 2, 0, 2, 0, 0, 0, 0, 0, 1, 1, 2, 2, 2, 0, 1, 2, 2,
       2, 2, 0, 0, 2, 2, 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2,
       2, 2, 2, 2, 1, 0, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0,
       1, 0, 1, 1, 0, 2, 0, 2, 1, 2, 1, 1, 1, 2, 2, 2, 2, 2, 2, 0, 2, 2,
       1, 1, 2, 2, 2, 1, 0, 0, 1, 2, 2, 1, 2, 2, 2, 0, 2, 1, 2, 2, 2, 2,
       1, 0, 2, 2, 2, 2, 1, 0, 1, 2, 2, 2, 1, 2, 1, 1, 2, 2, 0, 2, 2, 1,
       2, 1, 2, 1, 2, 0, 0, 2, 2, 1, 1, 1, 2, 2, 0, 0, 0, 1, 1, 1, 2, 0,
       1, 2, 0, 2, 0, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 0, 2, 1, 2, 0, 2, 2,
       2, 0, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 0, 2, 2, 1, 2, 2, 1, 0, 0, 2,
       2, 1, 2, 2, 2, 0, 0, 2, 2, 0, 0, 0, 0, 1, 1, 0, 0, 2, 2, 2, 2])
```

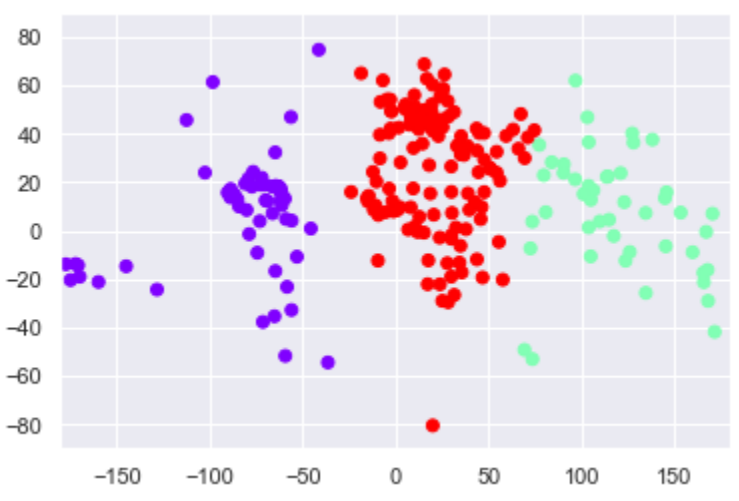
In [33]:
```
data_with_clusters = data.copy()
data_with_clusters['Cluster'] = identified_clusters
data_with_clusters.head()
```

Out[33]:

	name	Longitude	Latitude	Cluster
0	Aruba	-69.982677	12.520880	0
1	Afghanistan	66.004734	33.835231	2
2	Angola	17.537368	-12.293361	2
3	Anguilla	-63.064989	18.223959	0
4	Albania	20.049834	41.142450	2

Did you remember to use the c parameter to separate the data by the clusters we defined?

In [34]:
```
plt.scatter(data_with_clusters['Longitude'], data_with_clusters['Latitude'], c=data_w
plt.xlim(-180,180)
plt.ylim(-90,90)
plt.show()
```



If you haven't, go back and play around with the number of clusters.

Try 3, 7 and 8 and see if the results match your expectations!

In [57]:
```
data_less = pd.DataFrame(columns=['name','Longitude'])
```

In [59]:
```
data_less
```

Out[59]:

	name	Longitude

In [69]:
```
lista = []
for x in data['Longitude']:
    if x<-150:
        print(x)
        lista.append(x)
```

```
-170.7180258
-159.7872422
-169.8699468
-174.8098734
-177.3483483
-172.1648506
```

In [83]:
```
lista
type(lista)
```

Out[83]:
```
list
```

In [77]:
```
data_less['Longitude'].astype(float)
```

Out[77]:
```
Series([], Name: Longitude, dtype: float64)
```

In [86]:
```
lista_series = pd.Series(lista)
lista_series
```

Out[86]:
```
0   -170.718026
1   -159.787242
2   -169.869947
3   -174.809873
4   -177.348348
5   -172.164851
dtype: float64
```

In [95]:
```
data_less = data_less['Longitude'].append(lista_series)
```

In [96]:
```
data_less
```

Out[96]:
```
0   -170.718026
1   -159.787242
2   -169.869947
3   -174.809873
4   -177.348348
5   -172.164851
dtype: float64
```

In [ ]:
```

```