Hierarchical Agglomerative Clustering

In [1]:
```
# Import the standard modules
import pandas as pd
import numpy as np
%matplotlib inline
```

In [2]:
```
# Load the dataset
cust_pd = pd.read_csv("input/customers.csv")
```

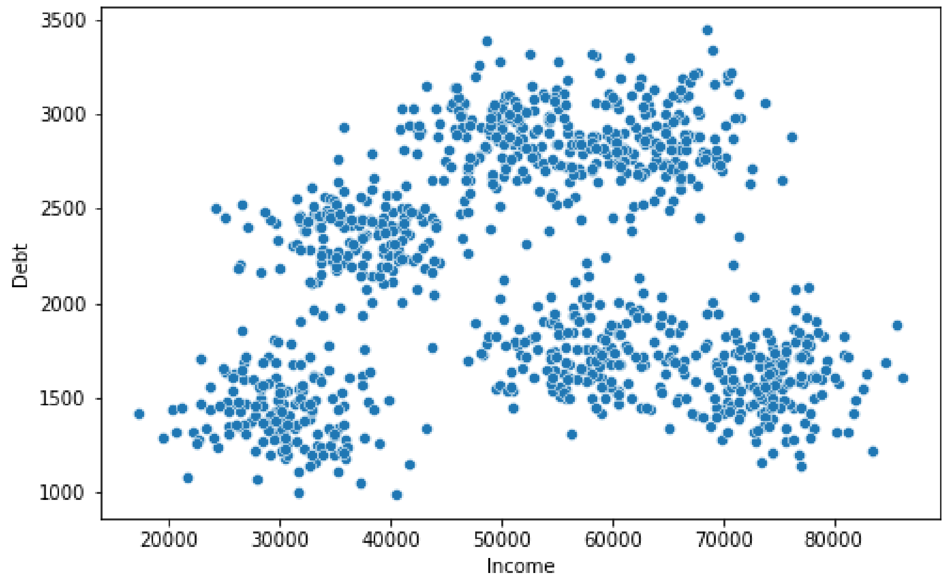
In [3]:
```
cust_pd.head()
```

Out[3]:

	Income	Debt
0	30309	1405
1	38969	1256
2	27268	1370
3	27970	1074
4	56432	1976

In [4]:
```
from matplotlib import pyplot as plt
import seaborn as sns
plt.figure(figsize=(8,5))
sns.scatterplot(x=cust_pd["Income"], y=cust_pd["Debt"])
```

Out[4]:
```
<AxesSubplot:xlabel='Income', ylabel='Debt'>
```



Normalization of Data

In [5]:
```
# Normalize the data [0-1] using MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
cust_pd = scaler.fit_transform(cust_pd)
cust_pd = pd.DataFrame(cust_pd, columns=["Income", "Debt"])
cust_pd.head()
```

Out[5]:

	Income	Debt
0	0.189918	0.169857
1	0.315582	0.109165
2	0.145790	0.155601
3	0.155977	0.035031
4	0.568985	0.402444

Creating a Dendrogram

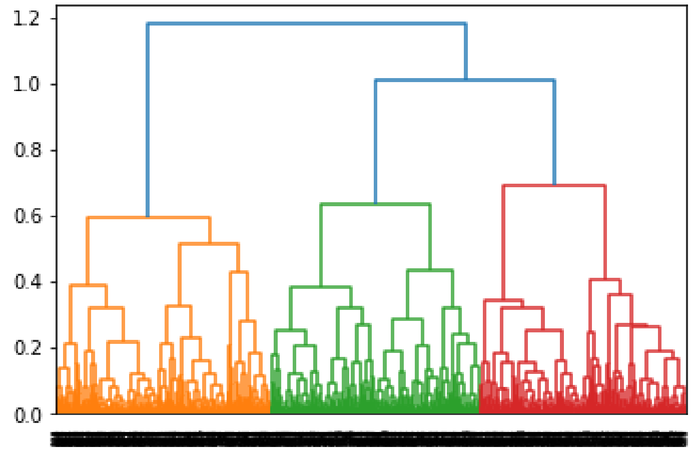
linkage method

1. method: str - single, complete, average, ward
2. metric: str - euclidean, cityblock, cosine, hamming (refer to scipy.spatial.distance.pdist) dendrogram method 1. orientation: str, optional

The direction to plot the dendrogram, which can be any of the following strings:

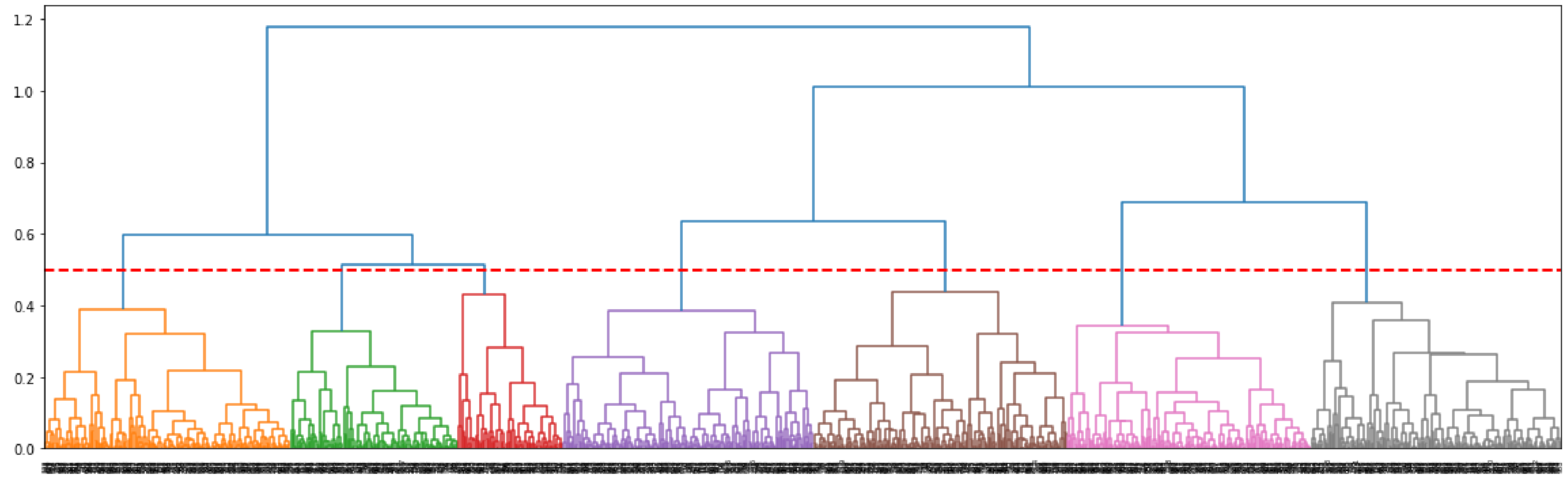
- 'top': Plots the root at the top, and plot descendent links going downwards. (default).
- 'bottom': Plots the root at the bottom, and plot descendent links going upwards.
- 'left': Plots the root at the left, and plot descendent links going right.
- 'right': Plots the root at the right, and plot descendent links going left.

In [6]:
```
# Create a dendrogram from scipy (to see the hierarchical relationship between, the clusters)
from scipy.cluster.hierarchy import dendrogram, linkage
linked = linkage(cust_pd, method='complete', metric='euclidean')
dendrogram(linked, orientation='top', distance_sort='ascending',show_leaf_counts=False)
plt.show()
```



We have created a dendrogram using complete linkage. As you can see, dendrogram partitions the data into three clusters, indicated by green, red and blue. We can specify the cut threshold so that data will be partitioned into the desired clusters. To specify the cut threshold, we pass a value to parameter color_threshold. All links connecting nodes with distances greater than or equal to the threshold will be colored blue. Let's specify 0.5 as the cut threshold.

In [7]:
```python
# Specify the cut threshold so that data will be partitioned into the desired clusters
plt.figure(figsize=(20,6))
dendrogram(linked, orientation='top', distance_sort='ascending',show_leaf_counts=False, color_threshold=0.5)
plt.axhline(y=0.5, c='r', linestyle='--', linewidth=2)
# we can draw a horizontal line to cut the dendrogram
plt.show()
```



As you can see the links with distances greater than or equal to 0.5 are colored blue. Seven clusters are created indicated by the green, red, light blue, purple, yellow, black and green. The clusters can also be seen based on cutting the horizontal line. Try using different linkages to see how the data will be clustered.

Clustering using Hierarchical Agglomerative Clustering

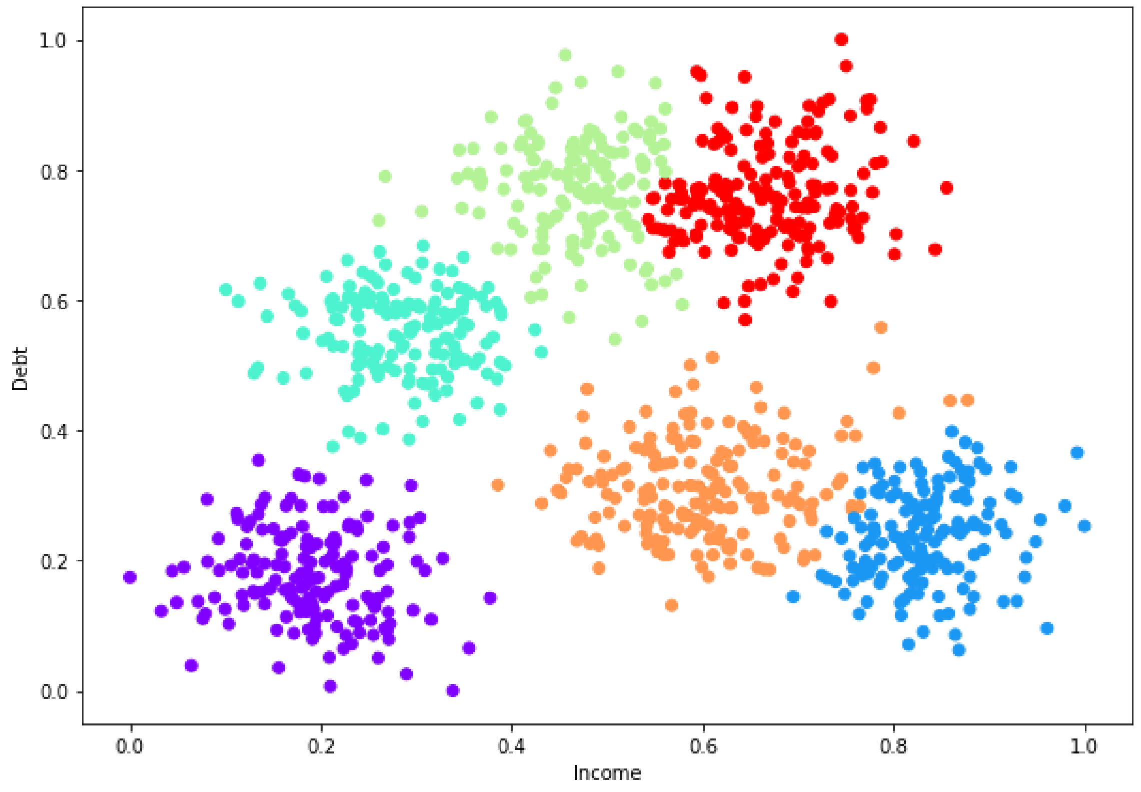
Let’s cluster the data using Agglomerative Clustering. We have to import the module from ScikitLearn. We specify six as number of clusters. Note that AgglomerativeClustering does not provide single linkage method.

Set parameters for AgglomerativeClustering, affnity refers to metric used to compute the linkage - **‘euclidean’**, **‘manhattan’**, **‘cosine’**. linkage criterion determines which distance to use between sets of observation - **‘ward’**, **‘complete’**, **‘average’**.

In [8]:
```
# Perform the customer dataset using Agglomerative Clustering
from sklearn.cluster import AgglomerativeClustering
model = AgglomerativeClustering(n_clusters=6, affinity='euclidean',linkage='ward') # by default ward linkage & euclidean metric
pred = model.fit_predict(cust_pd)
```

In [9]:
```
# Let's plot the clusters to see how actually our data has been clustered
plt.figure(figsize=(10, 7))
plt.scatter(x=cust_pd["Income"], y=cust_pd["Debt"], c=pred, cmap='rainbow_r')
plt.xlabel('Income')
plt.ylabel('Debt')
```

Out[9]:
```
Text(0, 0.5, 'Debt')
```



Based on the plot, we can see the data points in the form of six clusters. There are six clusters for the customer dataset based on the attribute Income and Debt indicated by the purple, green, light green, red, orange and blue.

```
In [ ]:
```