Unsupervised Learning

- Unsupervised Learning
  - 1.1. Supervised Learning and Unsupervised Learning
  - 1.2. Example and Application of Unsupervised Learning
    - 1.2.1. Application of Unsupervised Learning
    - 1.2.2. Clustering
    - 1.2.3. Learning Objective
    - 1.2.4. Clustering
    - 1.2.5. Types of Clustering
    - 1.2.6. Types of Clustering
    - 1.2.7. Hierarchical Clustering
    - 1.2.8. Working of Hierarchical Clustering
    - 1.2.9. Distance Measures
    - 1.2.10. The Dendrogram
    - 1.2.11. Example
    - 1.2.12. Clustering Animals
    - 1.2.13. Hierarchical Clustering Step 1
  - 1.3. Clustering Animals
  - 1.3.1. Clustering Animals
  - 1.3.2. Hierarchical Clustering Step 2
  - 1.3.3. Hierarchical Clustering Step 3
  - 1.3.4. Hierarchical Clustering Step 4
  - 1.3.5. Clustering Animals
  - 1.3.6. Plotting Hierarchical Clustering Dendrogram
  - 1.3.7. Plotting Hierarchical Clustering Dendrogram
  - 1.3.8. Plotting Hierarchical Clustering Dendrogram
  - 1.3.9. Plotting Hierarchical Clustering Dendrogram
  - 1.3.10. Plotting Hierarchical Clustering Dendrogram
  - 1.3.11. Plotting Hierarchical Clustering Dendrogram
  -

```
# if i want the ARM of the model
# Lasso regression for feature selection and fitting of the model
# Stepwise regression for feature selection
# fitting of the model
# fine tuning the model
# interpretation of the model
# plot the relationships
# Ridge regression for feature selection and fitting of the model
# Stochastic Gradient Descent
# plot the coefficients
```

Practice Project: Customer Segmentation

Customer segmentation is the practice of partitioning a customer base into groups of similar customers, where models developed for every group are expected to yield higher returns than those developed for the customer base as a whole. In this project, we will apply K-Means Clustering to a dataset of mall customers with their customer ID, gender, age, annual income and spending score (1-100). The company is curious to segment its customers into groups based on these attributes, so that it can customize its services accordingly.

Objective: Segment customers into different groups based on their shopping habits.

```
# Segmentation of customers based on their shopping habits
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans,AgglomerativeClustering,DBSCAN,MeanShift
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import scipy.cluster.hierarchy as shc
import os
os.chdir('C:/Users/Ashish/Desktop/Data_Sets')
data = pd.read_csv('Mall_Customers.csv')
data.shape
data.head()
data.info()
data.describe()
```

```
(200, 5)
   CustomerID  Genre  Age  Annual Income (k$)  Spending Score (1-100)
0           1   Male   19                  15                      39
1           2   Male   21                  15                      81
2           3 Female   20                  16                       6
3           4 Female   23                  16                      77
4           5 Female   31                  17                      40
     CustomerID  Genre  Age  Annual Income (k$)  Spending Score (1-100)
count  200.000    NaN  200          200.000000              200.000000
unique     NaN      2    NaN                NaN                     NaN
top        NaN   Male    NaN                NaN                     NaN
freq       NaN    112    NaN                NaN                     NaN
mean     100.5    NaN   38.85           60.560              50.200000
std       57.8    NaN   13.97           26.264              25.823522
min        1.0    NaN   18.00           15.000               1.000000
25%      100.5    NaN   28.75           41.500              34.000000
50%      200.5    NaN   36.00           61.500              50.000000
75%      300.5    NaN   49.00           78.000              73.000000
max      400.0    NaN   70.00          137.000              99.000000
<class'pandas.core.frame.DataFrame'>
RangeIndex:200 entries,0 to 199
Data columns(total 5 columns):
CustomerID                200 non-null int64
Genre                     200 non-null object
Age                       200 non-null int64
Annual Income (k$)        200 non-null int64
Spending Score (1-100)    200 non-null int64
dtypes:int64(4),object(1)
memory usage:7.8+KB
```

```
# Using Ward hierarchical clustering
# Calculate the distance between each pair of observations and updating across iterations
Ward = shc.linkage(data, method='ward')
Ward.shape
```

```
array([[  0.  ,  38.  ,   0.  ,   2.  ],
       [  1.  ,  39.  ,   0.  ,   2.  ],
       [  2.  ,  40.  ,   0.  ,   2.  ],
       [  3.  ,  41.  ,   0.  ,   2.  ],
       [  4.  ,  42.  ,   0.  ,   2.  ],
       [  5.  ,  43.  ,   0.  ,   2.  ],
       [  6.  ,  44.  ,   0.  ,   2.  ],
       [  7.  ,  45.  ,   0.  ,   2.  ],
       [  8.  ,  46.  ,   0.  ,   2.  ],
       [  9.  ,  47.  ,   0.  ,   2.  ],
       [ 10.  ,  48.  ,   0.  ,   2.  ],
       [ 11.  ,  49.  ,   0.  ,   2.  ],
       [ 12.  ,  50.  ,   0.  ,   2.  ],
       [ 13.  ,  51.  ,   0.  ,   2.  ],
       [ 14.  ,  52.  ,   0.  ,   2.  ],
       [ 15.  ,  53.  ,   0.  ,   2.  ],
       [ 16.  ,  54.  ,   0.  ,   2.  ],
       [ 17.  ,  55.  ,   0.  ,   2.  ],
       [ 18.  ,  56.  ,   0.  ,   2.  ],
       [ 19.  ,  57.  ,   0.  ,   2.  ],
       [ 20.  ,  58.  ,   0.  ,   2.  ],
       [ 21.  ,  59.  ,   0.  ,   2.  ],
       [ 22.  ,  60.  ,   0.  ,   2.  ],
       [ 23.  ,  61.  ,   0.  ,   2.  ],
       [ 24.  ,  62.  ,   0.  ,   2.  ],
       [ 25.  ,  63.  ,   0.  ,   2.  ],
       [ 26.  ,  64.  ,   0.  ,   2.  ],
       [ 27.  ,  65.  ,   0.  ,   2.  ],
       [ 28.  ,  66.  ,   0.  ,   2.  ],
       [ 29.  ,  67.  ,   0.  ,   2.  ],
       [ 30.  ,  68.  ,   0.  ,   2.  ],
       [ 31.  ,  69.  ,   0.  ,   2.  ],
       [ 32.  ,  70.  ,   0.  ,   2.  ],
       [ 33.  ,  71.  ,   0.  ,   2.  ],
       [ 34.  ,  72.  ,   0.  ,   2.  ],
       [ 35.  ,  73.  ,   0.  ,   2.  ],
       [ 36.  ,  74.  ,   0.  ,   2.  ],
       [ 37.  ,  75.  ,   0.  ,   2.  ],
       [ 38.  ,  76.  ,   0.  ,   2.  ],
       [ 39.  ,  77.  ,   0.  ,   2.  ],
       [ 40.  ,  78.  ,   0.  ,   2.  ],
       [ 41.  ,  79.  ,   0.  ,   2.  ],
       [ 42.  ,  80.  ,   0.  ,   2.  ],
       [ 43.  ,  81.  ,   0.  ,   2.  ],
       [ 44.  ,  82.  ,   0.  ,   2.  ],
       [ 45.  ,  83.  ,   0.  ,   2.  ],
       [ 46.  ,  84.  ,   0.  ,   2.  ],
       [ 47.  ,  85.  ,   0.  ,   2.  ],
       [ 48.  ,  86.  ,   0.  ,   2.  ],
       [ 49.  ,  87.  ,   0.  ,   2.  ],
       [ 50.  ,  88.  ,   0.  ,   2.  ],
       [ 51.  ,  89.  ,   0.  ,   2.  ],
       [ 52.  ,  90.  ,   0.  ,   2.  ],
       [ 53.  ,  91.  ,   0.  ,   2.  ],
       [ 54.  ,  92.  ,   0.  ,   2.  ],
       [ 55.  ,  93.  ,   0.  ,   2.  ],
       [ 56.  ,  94.  ,   0.  ,   2.  ],
       [ 57.  ,  95.  ,   0.  ,   2.  ],
       [ 58.  ,  96.  ,   0.  ,   2.  ],
       [ 59.  ,  97.  ,   0.  ,   2.  ],
       [ 60.  ,  98.  ,   0.  ,   2.  ],
       [ 61.  ,  99.  ,   0.  ,   2.  ],
       [ 62.  , 100.  ,   0.  ,   2.  ],
       [ 63.  , 101.  ,   0.  ,   2.  ],
       [ 64.  , 102.  ,   0.  ,   2.  ],
       [ 65.  , 103.  ,   0.  ,   2.  ],
       [ 66.  , 104.  ,   0.  ,   2.  ],
       [ 67.  , 105.  ,   0.  ,   2.  ],
       [ 68.  , 106.  ,   0.  ,   2.  ],
       [ 69.  , 107.  ,   0.  ,   2.  ],
       [ 70.  , 108.  ,   0.  ,   2.  ],
       [ 71.  , 109.  ,   0.  ,   2.  ],
       [ 72.  , 110.  ,   0.  ,   2.  ],
       [ 73.  , 111.  ,   0.  ,   2.  ],
       [ 74.  , 112.  ,   0.  ,   2.  ],
       [ 75.  , 113.  ,   0.  ,   2.  ],
       [ 76.  , 114.  ,   0.  ,   2.  ],
       [ 77.  , 115.  ,   0.  ,   2.  ],
       [ 78.  , 116.  ,   0.  ,   2.  ],
       [ 79.  , 117.  ,   0.  ,   2.  ],
       [ 80.  , 118.  ,   0.  ,   2.  ],
       [ 81.  , 119.  ,   0.  ,   2.  ],
       [ 82.  , 120.  ,   0.  ,   2.  ],
       [ 83.  , 121.  ,   0.  ,   2.  ],
       [ 84.  , 122.  ,   0.  ,   2.  ],
       [ 85.  , 123.  ,   0.  ,   2.  ],
       [ 86.  , 124.  ,   0.  ,   2.  ],
       [ 87.  , 125.  ,   0.  ,   2.  ],
       [ 88.  , 126.  ,   0.  ,   2.  ],
       [ 89.  , 127.  ,   0.  ,   2.  ],
       [ 90.  , 128.  ,   0.  ,   2.  ],
       [ 91.  , 129.  ,   0.  ,   2.  ],
       [ 92.  , 130.  ,   0.  ,   2.  ],
       [ 93.  , 131.  ,   0.  ,   2.  ],
       [ 94.  , 132.  ,   0.  ,   2.  ],
       [ 95.  , 133.  ,   0.  ,   2.  ],
       [ 96.  , 134.  ,   0.  ,   2.  ],
       [ 97.  , 135.  ,   0.  ,   2.  ],
       [ 98.  , 136.  ,   0.  ,   2.  ],
       [ 99.  , 137.  ,   0.  ,   2.  ],
       [100.  , 138.  ,   0.  ,   2.  ],
       [101.  , 139.  ,   0.  ,   2.  ],
       [102.  , 140.  ,   0.  ,   2.  ],
       [103.  , 141.  ,   0.  ,   2.  ],
       [104.  , 142.  ,   0.  ,   2.  ],
       [105.  , 143.  ,   0.  ,   2.  ],
       [106.  , 144.  ,   0.  ,   2.  ],
       [107.  , 145.  ,   0.  ,   2.  ],
       [108.  , 146.  ,   0.  ,   2.  ],
       [109.  , 147.  ,   0.  ,   2.  ],
       [110.  , 148.  ,   0.  ,   2.  ],
       [111.  , 149.  ,   0.  ,   2.  ],
       [112.  , 150.  ,   0.  ,   2.  ],
       [113.  , 151.  ,   0.  ,   2.  ],
       [114.  , 152.  ,   0.  ,   2.  ],
       [115.  , 153.  ,   0.  ,   2.  ],
       [116.  , 154.  ,   0.  ,   2.  ],
       [117.  , 155.  ,   0.  ,   2.  ],
       [118.  , 156.  ,   0.  ,   2.  ],
       [119.  , 157.  ,   0.  ,   2.  ],
       [120.  , 158.  ,   0.  ,   2.  ],
       [121.  , 159.  ,   0.  ,   2.  ],
       [122.  , 160.  ,   0.  ,   2.  ],
       [123.  , 161.  ,   0.  ,   2.  ],
       [124.  , 162.  ,   0.  ,   2.  ],
       [125.  , 163.  ,   0.  ,   2.  ],
       [126.  , 164.  ,   0.  ,   2.  ],
       [127.  , 165.  ,   0.  ,   2.  ],
       [128.  , 166.  ,   0.  ,   2.  ],
       [129.  , 167.  ,   0.  ,   2.  ],
       [130.  , 168.  ,   0.  ,   2.  ],
       [131.  , 169.  ,   0.  ,   2.  ],
       [132.  , 170.  ,   0.  ,   2.  ],
       [133.  , 171.  ,   0.  ,   2.  ],
       [134.  , 172.  ,   0.  ,   2.  ],
       [135.  , 173.  ,   0.  ,   2.  ],
       [136.  , 174.  ,   0.  ,   2.  ],
       [137.  , 175.  ,   0.  ,   2.  ],
       [138.  , 176.  ,   0.  ,   2.  ],
       [139.  , 177.  ,   0.  ,   2.  ],
       [140.  , 178.  ,   0.  ,   2.  ],
       [141.  , 179.  ,   0.  ,   2.  ],
       [142.  , 180.  ,   0.  ,   2.  ],
       [143.  , 181.  ,   0.  ,   2.  ],
       [144.  , 182.  ,   0.  ,   2.  ],
       [145.  , 183.  ,   0.  ,   2.  ],
       [146.  , 184.  ,   0.  ,   2.  ],
       [147.  , 185.  ,   0.  ,   2.  ],
       [148.  , 186.  ,   0.  ,   2.  ],
       [149.  , 187.  ,   0.  ,   2.  ],
       [150.  , 188.  ,   0.  ,   2.  ],
       [151.  , 189.  ,   0.  ,   2.  ],
       [152.  , 190.  ,   0.  ,   2.  ],
       [153.  , 191.  ,   0.  ,   2.  ],
       [154.  , 192.  ,   0.  ,   2.  ],
       [155.  , 193.  ,   0.  ,   2.  ],
       [156.  , 194.  ,   0.  ,   2.  ],
       [157.  , 195.  ,   0.  ,   2.  ],
       [158.  , 196.  ,   0.  ,   2.  ],
       [159.  , 197.  ,   0.  ,   2.  ],
       [160.  , 198.  ,   0.  ,   2.  ],
       [161.  , 199.  ,   0.  ,   2.  ],
       [162.  , 200.  ,   0.  ,   2.  ],
       [163.  , 201.  ,   0.  ,   2.  ],
       [164.  , 202.  ,   0.  ,   2.  ],
       [165.  , 203.  ,   0.  ,   2.  ],
       [166.  , 204.  ,   0.  ,   2.  ],
       [167.  , 205.  ,   0.  ,   2.  ],
       [168.  , 206.  ,   0.  ,   2.  ],
       [169.  , 207.  ,   0.  ,   2.  ],
       [170.  , 208.  ,   0.  ,   2.  ],
       [171.  , 209.  ,   0.  ,   2.  ],
       [172.  , 210.  ,   0.  ,   2.  ],
       [173.  , 211.  ,   0.  ,   2.  ],
       [174.  , 212.  ,   0.  ,   2.  ],
       [175.  , 213.  ,   0.  ,   2.  ],
       [176.  , 214.  ,   0.  ,   2.  ],
       [177.  , 215.  ,   0.  ,   2.  ],
       [178.  , 216.  ,   0.  ,   2.  ],
       [179.  , 217.  ,   0.  ,   2.  ],
       [180.  , 218.  ,   0.  ,   2.  ],
       [181.  , 219.  ,   0.  ,   2.  ],
       [182.  , 220.  ,   0.  ,   2.  ],
       [183.  , 221.  ,   0.  ,   2.  ],
       [184.  , 222.  ,   0.  ,   2.  ],
       [185.  , 223.  ,   0.  ,   2.  ],
       [186.  , 224.  ,   0.  ,   2.  ],
       [187.  , 225.  ,   0.  ,   2.  ],
       [188.  , 226.  ,   0.  ,   2.  ],
       [189.  , 227.  ,   0.  ,   2.  ],
       [190.  , 228.  ,   0.  ,   2.  ],
       [191.  , 229.  ,   0.  ,   2.  ],
       [192.  , 230.  ,   0.  ,   2.  ],
       [193.  , 231.  ,   0.  ,   2.  ],
       [194.  , 232.  ,   0.  ,   2.  ],
       [195.  , 233.  ,   0.  ,   2.  ],
       [196.  , 234.  ,   0.  ,   2.  ],
       [197.  , 235.  ,   0.  ,   2.  ],
       [198.  , 236.  ,   0.  ,   2.  ],
       [199.  , 237.  ,   0.  ,   2.  ],
       [200.  , 238.  ,   5.  ,   2.  ]])
(237, 4)
```

```
# Lets plot the dendrogram
# Lets cut the dendrogram at a height of 60
plt.figure(figsize=(6, 5))
plt.title("Customer Dendrograms")
dend = shc.dendrogram(Ward,truncate_mode='lastp')
cutoff = 60*np.max(dendrogram['dcoord'])
plt.plot(dendrogram['icoord'],dendrogram['dcoord'],'o')
plt.axhline(y=cutoff, c='k')
```

```
# Lets use the scikit-learn module
# Lets use the Euclidean distance metric
from sklearn.cluster import AgglomerativeClustering
# Lets use the ward linkage criterion
hclusterm = AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
clusters = hclusterm.fit_predict(data)
```

```
array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 2, 1, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3,
       2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3
```

Practice Project Image Segmentation

INSTRUCTIONS

This is an image market "sign-up." Our's looks cluttering with it on its arm hunter the image, which means that you want to keep the image's dimensions unchanged.

Objective: Open and display the image "sign.jpg." Convert the image into numpy array, in that it can be used in further processing. Find out the dimensions of the image and convert it into a two- dimensional array (Ex:n rows,n columns) for image segmentation, reducing the image to 10 clusters.

```
!wget https://www.dropbox.com/s/ldqqzqzqzqzqzqz/sign.jpg?dl=1
```

```
Sat Jun 12 00:00:00 2021
Usage: wget [OPTIONS]... [URL]...

Mandatory arguments to long options are mandatory.
  short options.

Startup:
  -V,  --version   display the version of Wget and exit.
  -h,  --help      print this help.
  -b,  --background  go to background after startup.
  -e,  --execute   execute a command.

Logging and input file:
  -o,  --output-file=FILE    log messages to FILE.
  -a,  --append-output=FILE  append messages to FILE.
  -d,  --debug               print lots of debugging information.
       --wdebug              print WEBDAV debug messages.
  -q,  --quiet               quiet (no output).
  -v,  --verbose             verbose (default).
  -nv, --no-verbose          turn off verbose without being quiet.
       --report-speed=X      output bandwidth as type X.
  -i,  --input-file=FILE     read URLs from FILE.
  -F,  --force-html          treat input file as HTML.
  -B,  --base=URL            resolve relative URL against URL.
       --sticky-dirs         keep to single directory.
       --unlink              remove file before clobber.

Download:
  -t,  --tries=NUMBER            set number of retrys to NUMBER (default: 20).
       --retry-connrefused       retry even if connection is refused.
       --retry-on-http-error=...
```

```
!ls
```

```
sample_data sign.jpg?dl=1
```

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from PIL import Image
img_path = 'sign.jpg?dl=1'
```

```
array([[255, 255, 255],
       [255, 255, 255],
       [255, 255, 255],
       [
```