Capstone Project Report

Name: Sumit Patel

Course: AI and ML (Batch – AUG 2020)

Duration: 10 months

Hierarchical K-Means: Construction of Hashing Tree

Problem Statement:

Factor analysis is a useful technique to find latent factors that can potentially describe multiple attributes, which is sometimes very useful for dimensionality reduction. Use the Airline Passenger Satisfaction dataset to perform factor analysis. (Use only the columns that represent the ratings given by the passengers, only 14 columns). Choose the best features possible that helps in dimensionality reduction, without much loss in information.

Prerequisites

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has latest version of python. The following url https://www.python.org/downloads/ can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-externalcommand/. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic. Second and easier option is to download anaconda and use its anaconda prompt to run the commands.

To install anaconda check this url https://www.anaconda.com/download/ You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run below commands in command prompt/terminal to install these packages:

```
pip install numpy

pip install pandas

pip install matplotlib

pip install sklearn

pip install scipy
```

If you have chosen to install anaconda then run below commands in anaconda prompt to install these packages:

```
conda install -c anaconda numpy

conda install -c anaconda pandas

conda install -c anaconda matplotlib

conda install -c anaconda sklearn

conda install -c anaconda scipy
```

Dataset used:

Dataset Link: Wholesale customers data https://archive.ics.uci.edu/ml/machine-learning-databases/00292/Wholesale%20customers%20data.csv

Importing the libraries and loading dataset.

```
# Import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram
```

Clustering the data into 2 clusters

```
model_1 = AgglomerativeClustering() # By default number of clusters = 2
model_1 = model_1.fit(x)
model_1.labels_
```

```
array([0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1,
       0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0],
      dtype=int64)
```

Modelling clusters based on Distance threshold

```
model_2 = AgglomerativeClustering(distance_threshold=0, n_clusters=None)
model_2.fit(x)
model_2.labels_
```

```
array([248, 293, 375, 240, 401, 427, 311, 379, 380, 252, 228, 378, 435,
       257, 347, 269, 360, 357, 264, 234, 368, 412, 439, 367, 249, 341,
       390, 350, 327, 325, 222, 372, 343, 261, 424, 279, 289, 221, 265,
       312, 333, 270, 315, 406, 300, 247, 397, 344, 391, 423, 316, 230,
       282, 299, 331, 260, 259, 241, 430, 281, 304, 335, 369, 171, 309,
       313, 362, 287, 238, 395, 292, 255, 305, 243, 314, 288, 409, 429,
       399, 363, 436, 183, 421, 431, 294, 271, 143, 329, 223, 245, 373,
       393, 277, 239, 275, 370, 402, 219, 434, 321, 353, 348, 374, 244,
       381, 146, 330, 278, 291, 383, 419, 415, 267, 276, 437, 138, 418,
       290, 236, 237, 354, 407, 394, 227, 326, 145, 416, 298, 420, 233,
       366, 358, 405, 388, 217, 411, 246, 253, 266, 306, 242, 155, 385,
       345, 113, 408, 119, 392, 318, 284, 382, 202, 209, 132, 203, 224,
       302, 414, 410, 340, 123, 332, 319, 250, 262, 258, 377, 226, 214,
       225, 220, 387, 336, 307, 339, 128, 134, 130, 324, 231, 346, 263,
       149, 111, 268, 396, 337,  72, 120, 232, 251, 356, 425, 121, 428,
       188, 317, 297, 301, 426, 254, 125, 213, 422, 285, 403, 432, 286,
       157, 376, 361, 156, 352, 184, 365, 116, 137, 195, 229, 389, 400,
       158, 272, 334, 204,  77, 283, 187, 141, 384, 338, 211, 417, 200,
       165, 172,  78, 256,  38, 310, 371, 168, 212, 162, 351, 131, 386,
       112,  59, 142, 207, 235, 182, 349, 320, 208, 438, 186, 154, 197,
       174, 127,  93, 190, 144, 205, 273, 117,  68, 328, 185, 166,  91,
       189, 176, 150, 194, 101, 364, 191, 210, 215, 280, 118, 359, 122,
       196, 355,  70, 274, 404,  65, 413, 178, 218, 322, 308, 398, 160,
       108, 159, 303, 296, 323, 163, 167, 140, 126, 124, 151, 115, 192,
       181, 433, 152, 169, 193, 295,  94, 139, 161, 206, 180, 179,  74,
        61, 106, 216, 114,  89,  83, 104,  58,  80,  79,  75, 164, 109,
       342,  82,  85, 103, 148,  57, 105,  56, 107, 102, 175,  63, 201,
```

Plotting Dendogram

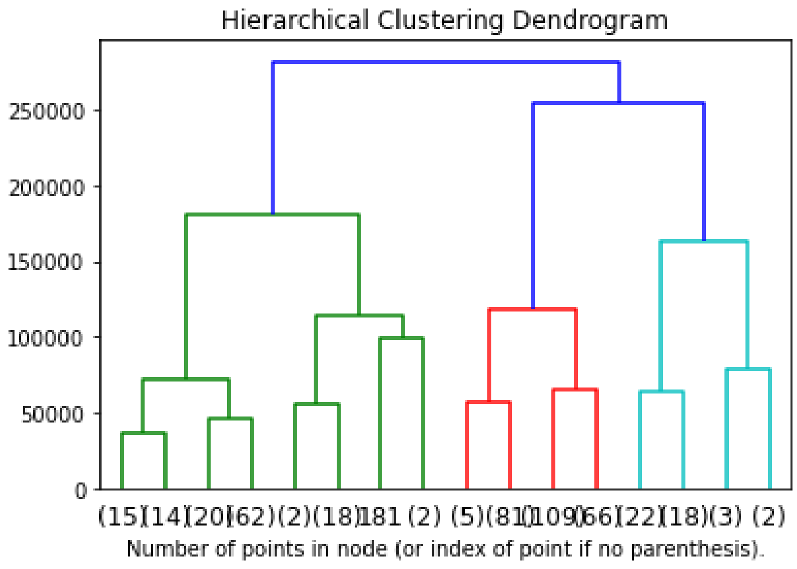
```
# Function to plot the dendrogram
def plot_dendrogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram

    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack([model.children_, model.distances_,counts]).astype(float)

    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs)
```

```
plt.title('Hierarchical Clustering Dendrogram')
# plot the top three levels of the dendrogram
plot_dendrogram(model_2, truncate_mode='level', p=3)
plt.xlabel("Number of points in node (or index of point if no parenthesis).")
plt.show()
```


Hierarchical Clustering Dendrogram
(15)(14)(20)(62)(2)(18)181 (2) (5)(81)(109)(66)(22)(18)(3) (2)
Number of points in node (or index of point if no parenthesis).