2020

HIERARCHICAL CLUSTERING TO PREDICT THE RESPONSE OF CARDIAC RESYNCHRONIZATION THERAPY IN PATIENTS WITH HEART FAILURE

Rukayat Bukola Adeosun
Michigan Technological University, radeosun@mtu.edu

HIERARCHICAL CLUSTERING TO PREDICT THE RESPONSE OF CARDIAC RESYNCHRONIZATION THERAPY IN PATIENTS WITH HEART FAILURE

By
Rukayat B. Adeosun

A REPORT
Submitted in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

In Health Informatics

MICHIGAN TECHNOLOGICAL UNIVERSITY

2020

This report has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Health Informatics.

Department of Applied Computing

Report Advisor: *Weihua Zhou*

Committee Member: *Guy Hembroff*

Committee Member: *Donald Peck*

Department Chair: *Daniel R. Fuhrmann*

Table of Contents

List of figures

List of tables

Acknowledgements

First, I would like to thank Almighty God for the strength and grace to complete this report. I also want to appreciate my parents, siblings and friends for their unwavering support all through my master's program. Finally, my appreciation to my advisor and graduate program director for all the help and guidance.

List of abbreviations

CRT - Cardiac Resynchronization Therapy

NYHA - New York Heart Agency

LBBB - Left Bundle Branch Block

SPECT - Single Photon Emission Computed Tomography

LV - Left Ventricle

LVEF - Left Ventricular Ejection Fraction

ESV - End Systolic Volume

IAEA - International Atomic Energy Agency

PBW - Phase BandWidth

CAD - Coronary Artery Disease

MI - Myocardial Infarction

ACEI - Angiotensin-converting enzyme inhibitors

ARB - Angiotensin Receptor Blocker

PSD - Phase Standard Deviation

QRS – Q wave, R wave and S wave that indicates ventricular depolarization

Abstract

The heterogeneous nature of today's evolving health databases requires new techniques and approaches to process these data and extract clinically useful information. This relevant information obtained can be used to improve the response rate of cardiac resynchronization therapy (CRT) in patients with heart failure. Hierarchical clustering (HC) which is an unsupervised ML technique may uncover clusters within the bulk of data of patient population which is useful for strategies towards precision and personalized medicine. This study aims to investigate how HC can be used to automatically group a bulk of clinically acquired CRT data into clusters and subgroups that could confer clinically relevant information. About 165 patient data were used in the study and the analysis resulted in 4 different phenogroups with varying response rates. Some features were statistically significant when compared within the subgroups. Lastly, the study concludes that HC can be used to integrate and analyze different kinds of clinical data to aid in the identification of HF patient subgroups that are likely to respond to CRT.

1. INTRODUCTION

Personalized medicine as a clinical model separates people into different groups. This classification aids doctors and researchers to predict and tailor accurate, efficient and highly effective treatments and prevention strategies to certain groups of patients with particular ailment. As opposed to using generalized treatments for all groups of people with certain diseases with little consideration to the differences in patients. Heart failure (HF) is a disease known to have differing signs and symptoms in a variety of patient groups with some treatments having little to no success. Cardiac resynchronization therapy (CRT) for HF patients is a therapy where individualized treatments for every unique patient could help overcome the limitations of traditional HF treatments.

1.1 Cardiac Resynchronization Therapy

CRT is a standard clinical treatment for a group of end-stage HF patients where a device sends little electrical charges to both lower chambers of the heart to help them beat together in a synchronized pattern which improves the heart's ability to pump blood and oxygen to the body [1]. The CRT procedure involves implanting a device the size of a pocket size watch just below the collar done. CRT is considered for HF patients who have explored correcting their heart condition through medication therapies without success. Although, CRT is suitable for HF patients who have moderate to severe symptoms with irregular heartbeat. It is not suitable for patients with diastolic heart failure, mild HF symptoms or patients whose left and right heart chambers beat synchronously.

1.1.1. Cardiac Resynchronization Therapy Devices

CRT devices help the heart beat more efficiently and help to monitor HF condition so that the physician can provide the right treatment. A CRT device has two main components which are the pulse generator and the thin insulated wires called leads which deliver a small amount of electrical energy to the heart to help restore the normal timing of the heart. The two types of CRT devices are the cardiac resynchronization therapy pacemaker (CRT-P) or biventricular marker and the same device with a built-in implantable cardioverter defibrillator called cardiac resynchronization therapy defibrillator (CRT-D) [1].

According to [2], CRT enhances outcome in patients with heart failure but also has substantial nonresponse rates. Several studies have shown improved CRT response prediction that includes varying criteria such as the Left Bundle Branch Block (LBBB), Left Ventricle (LV) activation time, intrinsicoid deflection onset as well as the frequency content and area of the Q wave, R wave and S wave (QRS). QRS shows the electrical impulse as it spreads through the ventricles and indicates ventricular depolarization.

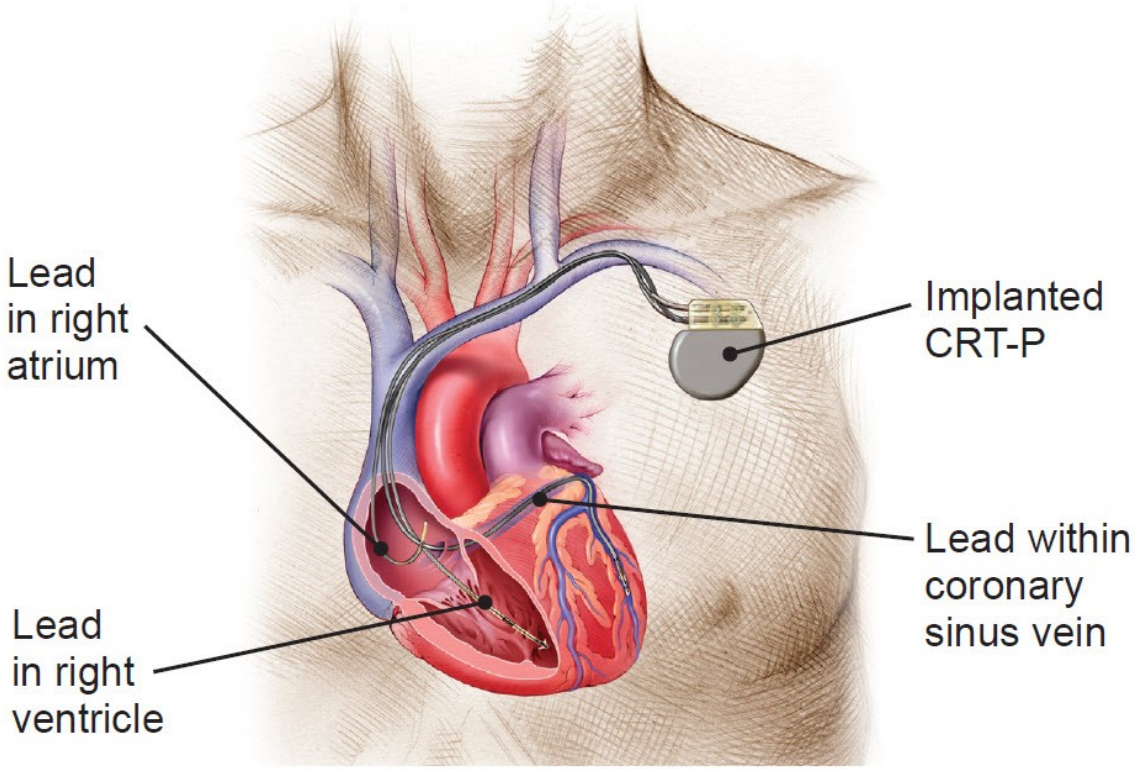

Figure 1: An implanted CRT-P system [1]

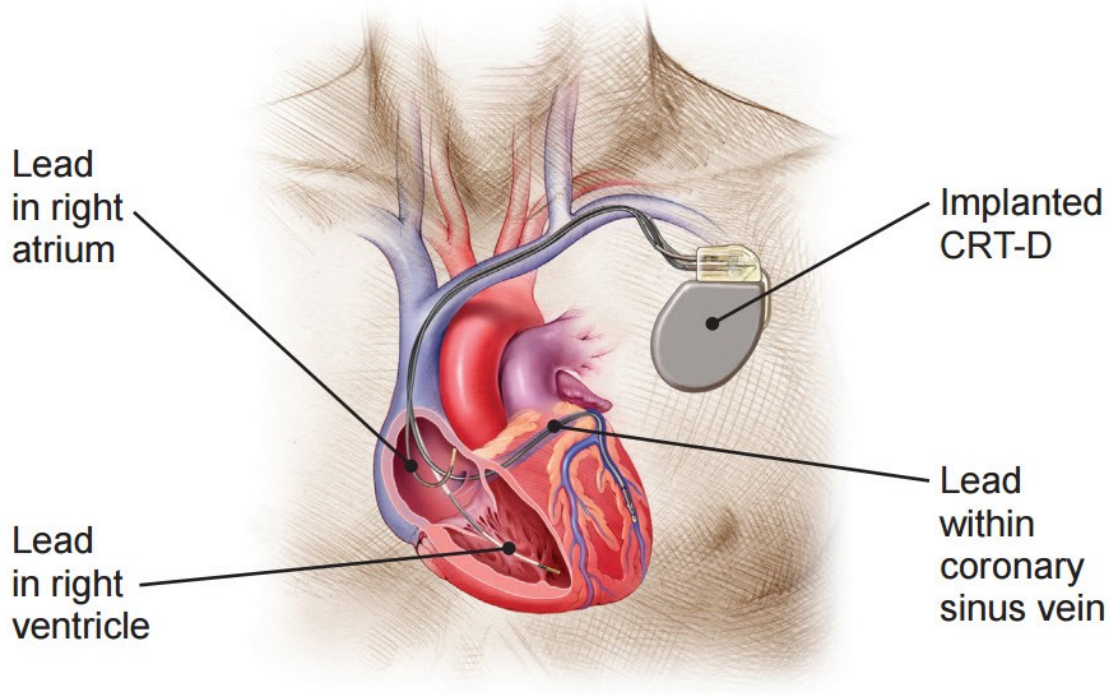

Figure 2: An implanted CRT-D system [1]

1.1.3 Cardiac Resynchronization Therapy Indications

The guidelines from the American Heart Association/American College of Cardiology Foundation (AHA/ACCF) for the management of HF which was published in 2013 were harmonized with the AHA/ACCF/Heart Rhythm Society (HRS) 2012 and referred to as the updated ACC/AHA/HRS guidelines [3]. The criteria for CRT implantation are New York Heart Association (NYHA) functional class II and III with sinus rhythm, Left ventricular Ejection Fraction (LVEF) < 35%, QRS width > 150 ms or 120 to 150 ms with Ventricular Electrical Dyssynchrony (ED) by LBBB. Although about 30% of patients do not respond to CRT due to common variables such as low LVEF, sinus rhythm, LBBB pattern, QRS duration >150ms on electrocardiogram (ECG) and the New York Heart Association (NYHA) class II, III, and ambulatory IV symptoms. This study also includes not only the Electrical Dyssynchrony (ED) data, which would be QRS enlargement (> 150 ms) and LBBB but also the presence of Mechanical Desynchronization (MD).

1.2. Machine Learning

The current trend in data analysis is towards technical approaches such as Machine Learning (ML) and even more powerful techniques like deep learning which uses neural networks to solve complex problems. However, deep learning requires a large set of data and this volume of data is often lacking in clinical medicine to aid better clinical predictions [4]. ML is a part of artificial intelligence that gives computer systems the ability to learn automatically with little to no human intervention and it adjusts its actions accordingly without being explicitly programmed. Machine learning improves the analysis of large quantities of data and is being used in a variety of applications like predictive analytics, email filtering and computer vision. Awan et al [5] discussed the application of machine learning methods in diagnosis, classification, readmissions and medication adherence in patients with heart failure. Also, ML techniques have been applied to identify distinct phenogroups in HF patients with preserved ejection fraction (HFpEF) as well as several diseases [6,7]. Thus, machine learning approaches may be used to improve CRT response prediction in patients with heart failure.

Supervised ML is a kind of ML that learns from the analysis of a known training dataset and then uses the learned labels to predict future events or analysis. While supervised ML provides targets for any new input after sufficient training, it requires massive dataset to train on, interpretation of result varies and it is susceptible to errors. In contrast, unsupervised ML requires no training or labelling as it explores the unlabelled data and can draw inferences from datasets to describe hidden structures [4]. Unsupervised ML groups or clusters patients together based on multiple characteristics in the dataset, which could be clinical, demographic, or measured. The grouping of similar patients together in varying groups or dimensions, then makes it possible to analyse the similarities in the groups of patients and relate them to clinical outcomes or therapeutic responses.

1.2.1. Hierarchical Clustering

Hierarchical clustering (HC) works by grouping data objects into a tree of clusters and HC can be further divided into agglomerative and divisive which simply refers to the hierarchical decomposition being formed bottom-up or top-down respectively. HC technique is a good way to reveal anatomical subgroups from clinical data as they do not need any prior information about the population of study. Also, HC does not require conditioning of an expected number of subgroups in contrast to K-means clustering that involves specifying a required number of subgroups. Some studies have applied HC techniques to 3D patient shape data and the outcome depends hugely on the clustering distance metrics and linkage option of choice [8].

Furthermore, HC results are graphically depicted in a tree-like structural diagram called a dendrogram that shows how similar objects are grouped together, while dissimilar objects are grouped on different branches of the tree. However, assessment of the similarity or dissimilarity and clustering results of objects is dependent on the similarity or distance metric chosen where low inter-object distance connotes high similarity. Also, the linkage function is another huge determinant on how objects are linked together to form a subgroup. Choosing the appropriate distance or linkage combination is necessary to achieve meaningful results.

1.3 Aim and Significance of Study

An area where clustering of patient groups could improve the selection of patients and accurately predict their clinical outcome is in CRT for HF patients. This is due to the fact that despite clear criteria for which patients should undergo CRT, a significant percentage of patients do not respond to this treatment option [9]. This study hypothesized that new approaches based on unsupervised ML techniques which incorporates demographic, clinical and Echocardiographic (ECG) and SPECT MPI data including both electrical and mechanical dyssynchrony may be used to better depict how ML can be utilized to phenogroup HF patients in correlation to their characteristics and predict clinical response.

Supervised ML mainly trains a model on a labelled dataset where the input data is split into training and test dataset and the algorithm learns to predict the output from the input data. Thus, unsupervised ML is used for this study instead of supervised ML because our aim is to identify hidden patterns or underlying grouping structure in our data in order to better predict HF patients' subgroups that are likely to respond to CRT. We utilized data from ten centres in 8 countries with NYHA functional class II, III and IV with sinus rhythm, LVEF < 35%, QRS width > 150 ms or 120 to 150 ms with Ventricular Electrical Dyssynchrony (ED) by Left Bundle Branch Block (LBBB) to determine if HC as an unsupervised ML technique could help discover unknown patterns in the data and to identify the patient subgroups that are more likely to respond to CRT.

2. RELATED WORK

Several studies have tried to find parameters or factors that could significantly improve CRT efficacy. The use of SPECT images to assess LV latest activation was shown in [10] to improve the rate of placing LV on target and ultimately produced a positive improvement in CRT response. Other studies have used machine learning approaches to predict mortality in patients with coronary artery disease. Cikel et al [4] researched the use of ML to accurately phenogroup selected CRT patients to determine trends that can result in improved CRT response. The study used unsupervised learning methods to help in the identification of patients likely to respond to the therapy by integrating clinical features with echocardiographic data on myocardial infarction and left ventricular volume changes that were measured over an entire cardiac cycle.

The results from the study showed that full unsupervised ML techniques can provide a clinically relevant classification of a heterogeneous group of HF patients which can aid the identification of patient subgroups most likely to respond to particular therapies. However, the paper reiterated that the feasibility of the proposed model for phenogrouping HF patients and in clinical decision making should be assessed in a prospective controlled trial. Additionally, Matthew et al [9] also used machine learning-based unsupervised clustering analysis to identify clinically distinct phenotypic subgroups in a highly dimensional mixed-data group of individuals with heart failure with preserved ejection fraction (HFpEF).This study was able to identify phenogroups of HFpEF patients with distinct clinical characteristics and lasting outcomes.

3. METHODS

3.1 Study Design and Patient Population

The population used for this study has been previously used by a prospective, multicenter, non-randomized trial: ‘Value of intraventricular synchronism assessment by gated-SPECT myocardial perfusion imaging in the management of heart failure patients submitted to cardiac resynchronization therapy’ (IAEA VISION-CRT) [11]. In brief, the VISION-CRT trial involves subjects from ten centers in 8 countries (Brazil, Chile, Colombia, Cuba, India, Mexico, Pakistan, and Spain). The main investigators of the respective countries recorded all the clinical, CRT and follow-up information in individual forms for each patient. The data from the Myocardial Perfusion Imaging by Gated Single Photon Emission Computed Tomography (gSPECT MPI) were recorded too. The overall data was collected by the central management center in the IAEA headquarters in Vienna. The subjects underwent a detailed clinical and gated SPECT MPI evaluation before recruitment to the study and all patients provided written informed consent.

The criteria for patient inclusion were: symptomatic HF patients over 18 years old with NYHA functional class II, III or ambulatory IV with HF for at least three months before enrollment; LV ejection fraction ≤ 35% from ischemic or non-ischemic causes, measured according to the usual

procedure at the participating center for inclusion, although LVEFs used for analysis came from nuclear core lab; sinus rhythm with LBBB configuration defined as a wide QRS duration ≥ 120ms. The Exclusion criteria were: pregnancy or breast-feeding; arrhythmias that prevented the gated acquisition; right bundle branch block; a major coexisting illness affecting survival less than one year; acute coronary syndromes, coronary artery bypass grafting, or percutaneous coronary intervention in the last 3 months before enrollment and within 6 months of CRT implantation. The patients were classified as 'responders' to CRT if they had an increase of LVEF > 5% or a decrease in End Systolic Volume (ESV) < -15% as measured by gated SPECT MPI at follow up. Others were classified as non-responders.

3.2. SPECT MPI evaluation

Gated SPECT scans were performed about 30 minutes after injection using 20-30mCi of 99mTc-sestamibi or tetrofosmin of 740 to 1110 MBq. The images were acquired in gamma cameras using 180° orbits with a complimentary 8 or 16 frames ECG-gating. The Ordered Subset Expectation maximization (OSEM) method with three iterations and ten subsets and filtered by a Butterworth filter to the power of 10 and a cut-off frequency of 0.3 cycles/mm were used to reconstruct the images and this was done by Emory Reconstruction Toolbox (ERToolbox; Atlanta, GA). The resulting reoriented short-axis images were sent to Emory Cardiac Toolbox (ECTb4, Atlanta, GA) for automated accessing of LV function, including LVEF, left ventricular end-systolic volume (LVESV), left ventricular end-diastolic volume (LVEDV), LV shape, including end-systolic eccentricity (ESE) and end-diastolic eccentricity (EDE), and LV mechanical dyssynchrony and includes phase standard deviation (PSD) and phase bandwidth (PBW) [11].

3.3. Baseline Characteristics

The complete data of clinical assessment, baseline SPECT MPI, and clinical six-month follow-up data were obtained in only 177 patients out of the initial 199 patients that underwent CRT. About 11 patients among the 177 patients died before follow-up and 1 patient had an extremely low ESV which is an outlier caused by the low resolution of gated SPECT MPI when measuring a small heart. This study finally utilized the data from 165 patients for its analysis. The covariates consist of a range of domains including demographics, clinical variables, laboratory data, SPECT MPI measurements and an electrocardiographic parameter. Overall, a total of 26 continuous and categorical variables were used in the clustering analysis.

3.4. Characterization of phenogroups

Agglomerative hierarchical clustering algorithm was used to group similar objects into clusters such that each observation starts in its own cluster and pairs of clusters are merged as one moves

up the hierarchy. The hierarchical relationship between the different set of data is shown in a tree-like diagram called a dendrogram. Furthermore, the distance between the data points on the x-axis represents the dissimilarities between the points while the height of the blocks on y-axis represents the distance between the clusters. Of the three most common linkage methods: single, complete and average linkage methods, the complete linkage method was used to merge the clusters in the dendrogram as it tends to find compact clusters of approximately equal diameters. The complete linkage method also avoids the disadvantage of the alternate single linkage method where clusters are forced together due to single objects being close to each other, even when many of the objects in each cluster may be largely distinct to each other. The number of clusters was chosen by drawing a horizontal line to the longest line that traverses the maximum distance up and down without intersecting the merging points. This was done on our dendrogram at both a distance 600 which gave 2 clusters and a distance of 400 which gave 4 clusters and the respective numbers of clusters were analyzed separately. Having done the analysis, 4 clusters were finally used for further analysis as it was more statistically significant.

After the clusters were grouped into four, the differences in demographics, clinical variables, laboratory data, SPECT MPI measurements, and echocardiographic parameters were compared between the phenogroups. Continuous variables were summarized in means and standard deviations while categorical variables were summarized in numbers and percentages. The differences between groups were tested using a one-way ANOVA for continuous variables and a Chi-squared test for categorical variables. A p-value of <0.05 was considered statistically significant. The resulting dendrogram was internally validated by shuffling the dataset and reducing the number of rows and columns to create clusters and compare their differences in terms of clinical characteristics and the response outcome. Clustering the phenogroup into 4 clusters was more statistically significant and the ML algorithm used for all the analysis in this study was done in python version 3.

```
# Calculate the distance between each sample
# linkage method defines how the distance between clusters is measured
# In complete linkage, the distance between clusters is the distance between the furthest points of the clusters
# In single linkage, the distance between clusters is the distance between the closest points of the clusters.

# Perform the necessary imports
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt

# Calculate the linkage: mergings
mergings = linkage(df1, method='complete')

# Plot the dendrogram, using varieties as labels
plt.figure(figsize=(35,15))
dendrogram(mergings,
           labels=df1.index.tolist(),
           leaf_rotation=90,
           leaf_font_size=8,color_threshold=240)
plt.show()
```

```
# Control number of clusters in the plot + add horizontal line.
plt.figure(figsize=(35,15))
dendrogram(mergings,
           labels=df1.index.tolist(),
           leaf_rotation=90,
           leaf_font_size=8,color_threshold=240)
plt.axhline(y=400, c='black', lw=1, linestyle='dashed')
plt.show()
```

Figure 3: Code snippets of Hierarchical Clustering

Table 1. Baseline characteristics of the study patients by phenogroups

S/N	FEATURES	OVERALL AVERAGE (meanSD) (count(%)) N=165 (98R,67NR) **59%R**	PHENO-GROUP ONE n=42 (21R,21NR) **50%R**	PHENO-GROUP TWO n=81 (55R,26NR) **68%R**	PHENO-GROUP THREE n=16 (10R, 6NR) **63%R**	PHENO-GROUP FOUR n=26 (12R,14NR) **46%R**	GROUP P-VALUE
1	ACEI_or_ARB	136(82%)	29(69%)	72(89%)	14(88%)	21(81%)	**0.049**
2	Age	60±11	61±10	61±12	58±10	58±10	0.1
3	CAD	51(31%)	16(38%)	23(28%)	2(13%)	10(38%)	0.213
4	Concordance	40(25%)	12(29%)	16(20%)	6(38%)	6(23%)	0.412
5	DM	41(25%)	8(19%)	22(27%)	6(38%)	5(19%)	0.424
6	ECG_pre_QRSd	161±25	156±23	159±24	161±28	176.6±21	**0.00092**
7	Gender	M=98(59%) F=67(41%)	M=31(74%) F=11(26%)	M=33(41%) F=48(59%)	M=13(81%) F=3(19%)	M=21(81%) F=5(19%)	**0.0**
8	HTN	97(59%)	29(69%)	40(49%)	11(69%)	17(65%)	0.116
9	LBBB	165(100%)	42(100%)	81(100%)	16(100%)	26(100%)	1.0

10	MI	35(21%)	11(26%)	12(15%)	2(13%)	10(38%)	**0.047**
11	NYHA	II=46(28%) III=101 (61%) IV=18 (11%)	II=12(29%) III=27(64%) IV=3(7%)	II=28(35%) III=47(58%) IV=6(7%)	II=3(19%) III=9(56%) IV=4(25%)	II=3(12%) III=18(69%) IV=5(19%)	0.094
12	Race I = Africa II = Asia III = Caucasian IV = Hispanic V = Indian	I=17(10%) II=6(4%) III=23(14%) IV=87(53%) V=32(19%)	I=6(14%) II=1(2%) III=5(12%) IV=24(57%) V=6(14%)	I=4(5%) II=3(4%) III=14(17%) IV=37(46%) V=23(28%)	I=2(13%) II=2(13%) III=1(6%) IV=8(50%) V=3(19%)	I=5(19%) II=0(0%) III=3(12%) IV=18(69%) V=0(0%)	**0.033**
13	SPECT_pre_EDE	0.5±0.2	0.5±0.1	0.6±0.2	0.5±0.2	0.5±0.2	0.17
14	SPECT_pre_EDSI	0.8±0.1	0.8±0.1	0.8±0.1	0.9±0.1	0.9±0.1	0.13
15	SPECT_pre_EDV	257.6±105	277.8±38	176±44	476.3±55	347.1±30	**<0.001**
16	SPECT_pre_ESE	0.6±0.2	0.6±0.2	0.6±0.2	0.4±0.2	0.5±0.1	**0.009**
17	SPECT_pre_ESSI	0.8±0.1	0.8±0.1	0.8±0.1	0.9±0.1	0.9±0.1	**0.004**
18	SPECT_pre_ESV	192.7±96	209.2±28	117.4±37	396±62	275.2±27	**<0.001**
19	SPECT_pre_LVEF	27.7±10.3	24±7.0	34.1±9.5	17.1±5.1	20.6±6.0	**0.003**
20	SPECT_pre_PBW	152.4±73.8	191.7±66.3	106.5±49.7	239.9±47.9	178.2±64.9	0.24
21	SPECT_pre_PSD	48.8±19.7	56.7±15.6	37.6±15.7	72.9±15.2	55.8±17	0.09
22	Smoking	27(16%)	7(17%)	15(19%)	2(13%)	3(12%)	1.0
23	SPECT_pre_50scar	25±14.4	29.2±13.6	17.8±11.5	35.1±15	31±14.3	0.827

24	LVEF	25±6.0	24.5±4.4	27.2±5.6	20.4±5.4	22±6.7	**0.005**
25	Echo_pre_EDV	192±36.8	193.3±17	180.7±33	195.3±10	224.4±30	**0.0002**
26	Echo_pre_ESV	149±37.9	155.3±21.6	132.3±30.8	157±18.3	187.7±53.5	**0.00006**

4. RESULTS

Of the 165 patient data used in this unsupervised learning study, there was a 59% response rate where 98 participants responded to CRT and 67 patients had no response. The race population was 53% hispanics, 19% indians, 14% caucasian, 10% african and 4% asians. The baseline characteristics of the study population are shown in Table 1. For all the patients, the age was 60 ± 11 years, and 98 (59.4%) patients were male. Fifty-one (30.9%) patients had a previous history of CAD while 97 (58.8%) had Hypertension (HTN). About 27(16%) are smokers while 41 (24.8%) had diabetes mellitus (DM). Although the study data has NYHA class II, III and IV, NYHA class III was predominant in the data with a rate of 61%. Myocardial infarction was not prevalent among the participants as it was present in only about 21%.

There were no statistically significant differences in age and the NYHA class distributions across phenogroups. Participants in phenogroup 1 had the lowest mean of ECG_pre_QRSd and a high number and rate of HTN as compared with the other groups. Phenogroup 2 had the highest burden of DM and the most significant response rate in females. It is also the group with the highest response rate and the least rate of hypertensive patients. While phenogroup 3 had the least burden of CAD and MI and the highest rate of DM. Phenogroup 4 had the least responders to CRT, the highest rate of NYHA class III patients, and the largest ECG_pre_QRSd mean as compared with the other groups. While phenogroup 1 and 4 had the same intermediate rates of both CAD at 38% and DM at 19%, phenogroup 1 and 3 had the same intermediate burden of HTN at 69%.

Among SPECT MPI parameters, phenogroup 1 participants had an intermediate high mean values for LVEF, SPECT_pre_LVEF, SPECT_pre_PSD and SPECT_pre_PBW as compared with the other phenogroups.While phenogroup 2 had the least mean values for SPECT_pre_EDV, SPECT_pre_ESV and SPECT_pre_50scar but has the highest mean and standard deviation of SPECT_pre_LVEF when compared with the other groups. Phenogroup 3 had the highest means for SPECT_pre_PBW, SPECT_pre_PSD and SPECT_pre_50scar values. The highest mean and standard deviation of SPECT_pre_ESV values are found in this group as well. Finally, the highest mean and standard deviation of Echo_pre_ESV and Echo_pre_EDV were in phenogroup 4.

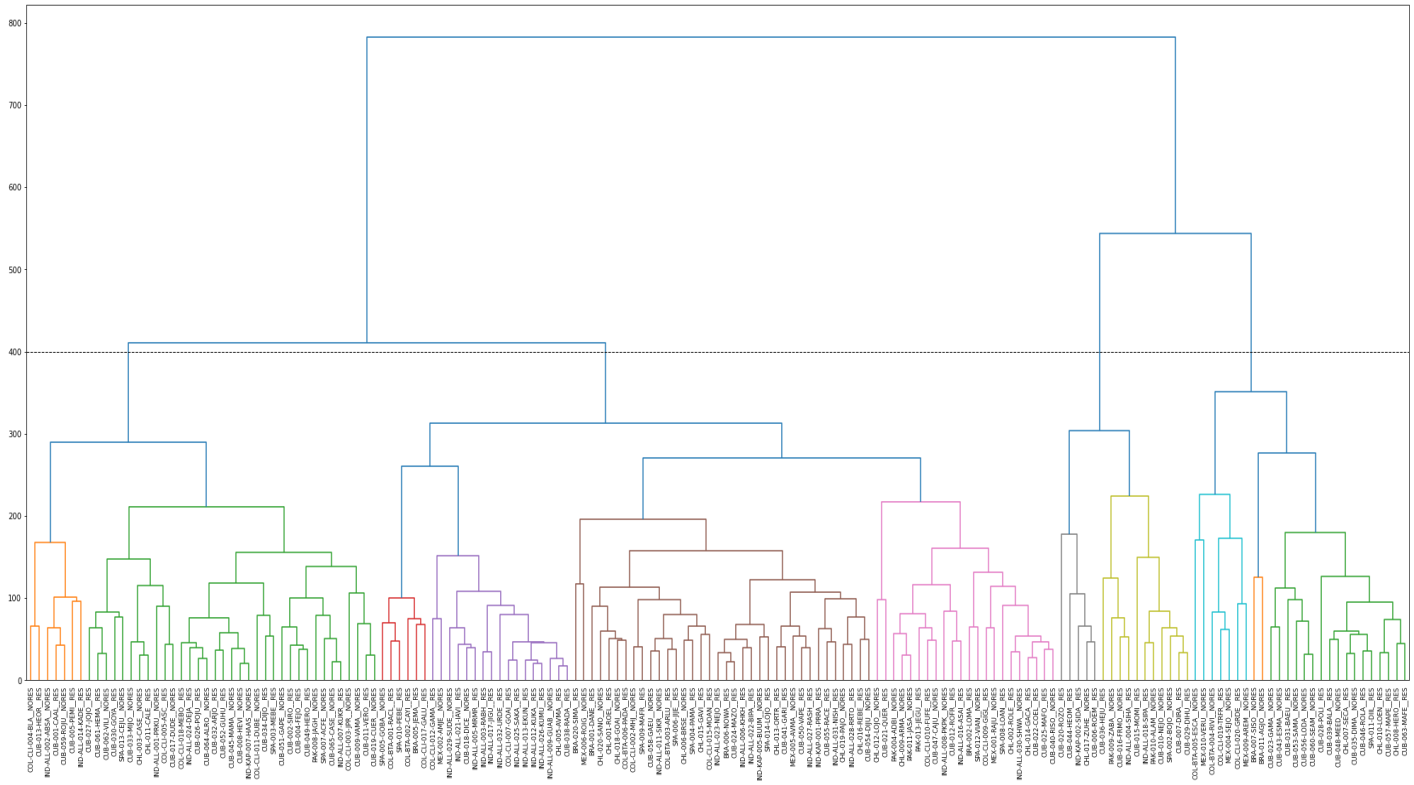

Figure 4: Hierarchical Clustering Dendrogram

Table 2: Dissimilarity between phenogroups

S/N	Phenogroup 1	Phenogroup 2	Phenogroup 3	Phenogroup 4
	- Least mean of ECG_pre_QRSd - Intermediate high values for LVEF, SPECT_pre_LVEF SPECT_pre_PSD, SPECT_pre_PBW	- Highest responder group - High proportion of patients with Diabetes Mellitus - Most significant response rate from females - Least rate of hypertensive patients - Has the highest mean and a wider range of SPECT_pre_LVEF - Least mean of SPECT_pre_EDV and SPECT_pre_ESV values - Least mean and standard deviation of	- Has the highest mean and a wider range of SPECT_pre_EDV values - Has lowest rate of NYHA class III patients - Has the highest mean and a wider range of SPECT_pre_ESV values - Has the highest mean of SPECT_pre_PSD and SPECT_pre_50scar values	- Has least responders to CRT - Has longest QRS duration - Has highest rate of NYHA class III patients - Has a high SPECT_pre_50scar mean and the highest standard deviation - Has the highest mean of Echo_pre_EDV values

		SPECT_pre_50scar values		- Has the highest mean and standard deviation of Echo_pre_ESV values

5. DISCUSSION

Heart failure is a disease characterized by multiple syndromes and its response to therapies is based on a couple of factors such as biomarkers, clinical data as well as imaging parameters. Conventional techniques to predict outcomes within HF subgroups rely on isolated parameters such as QRS morphology, presence or absence of specific comorbidities, HF cause, cardiac structure and function amongst others. Thus, while the use of echocardiographic analysis tools to assess cardiac structure and function can establish subgroups of HF patients at higher risk for negative outcomes [12], echocardiographic data contains a lot of information representing several time points in a cardiac cycle but this is replaced by single measurements in standard quantitative data analysis which does not take into account the complex events of the entire cardiac cycle.

In this study, hierarchical clustering as a form of unsupervised ML has been shown to aid the integration of demographic data, clinical data, laboratory data and SPECT MPI parameters to group patients with certain diseases such as HF. The research demonstrates the value of combining different sets of descriptors to find patients that are more likely to respond to CRT as compared to the results gotten from independent analysis of clinical parameters only. The results from the study proves that unsupervised ML approaches can be used to combine standard clinical parameters, ECG data and imaging parameters to provide a clinically interpretable and meaningful classification of the heterogenous phenotypes of HF patients and the likelihood of patients in certain subgroups to respond to specific treatment therapies.

While studies such as Chung et al [13] tried to find a single echocardiographic measure of dyssynchrony to improve the selection of HF patients for CRT beyond the current guidelines without success. This study combines heterogeneous data in an unsupervised manner to ultimately find groups of patients with similar characteristics towards CRT response. The unsupervised ML method used allows the natural clustering of patients and results in the identification of patient subgroups in relation to their CRT response. Specifically, Phenogroups 2 and 3 had a higher rate of response at 68% and 63% respectively over the overall rate at 59%. Phenogroups 1 and 4 had the least response rate at 50% and 46% respectively. About 14 parameters were statistically significant when compared within the 4 phenogroups. SPECT_pre_PSD, Echo_pre_ESV, Echo_pre_EDV, LVEF, SPECT_pre_PBW, MI,

SPECT_pre_LVEF, SPECT_pre_ESV, SPECT_pre_ESSI, SPECT_pre_ESE, SPECT_pre_EDV, ECG_pre_QRSd, gender and ACEI_or_ARB.

Some limitation of this study includes the small size used as well as the fact that HC does not work well with missing data. Though, the race parameter was significant in this study, this cannot be ascertained as the data was majorly from North America, South America and Asia. This means the data may be skewed towards a certain race over others and this is another limitation of this study. Furthermore, the results gotten from this study need to be externally validated. For future directions, further analysis using other unsupervised techniques such as principal component analysis may be able to ascertain the result or to help uncover other relevant clinical information. Another future work is the use of supervised classification to validate the result gotten in this study.

6. CONCLUSION

This study concludes that unsupervised ML approaches such as HC can be used to integrate and analyze ECG data, imaging parameters and clinical data to aid in the identification of HF patients subgroups that are likely to respond to CRT. The results show that HC can provide a clinically relevant classification of a heterogeneous cohort of HF patients which can serve as a data-driven basis to identify patient phenogroups likely to respond to specific therapies. However, the feasibility of this HC approach for patient phenogrouping in HF and its contribution to clinical decision making needs to be evaluated with a large dataset, externally validated and used in a prospective controlled trial.

7. REFERENCES

1) https://www.bostonscientific.com/en-US/patients/about-your-device/crt-devices/how-crts-work.html.
2) *Albert K. Feeny, John Rickard, Kevin M. Trulock, Divyang Patel, Saleem Toro, Laurie Ann Moennich, Niraj Varma, Mark J. Niebauer, Eiran Z. Gorodeski, Richard A. Grimm, John Barnard, Anant Madabhushi, Mina K. Chung*. Machine Learning of 12-Lead QRS Waveforms to Identify Cardiac Resynchronization Therapy Patients With Differential Outcomes. Arrhythmia and Electrophysiology. 2020;13
3) *Camilla Normand, Cecilia Linde, Jagmeet Singh, Kenneth Dickstein*. Indications for Cardiac Resynchronization Therapy: A Comparison of the Major International Guidelines, JACC: Heart Failure, Volume 6, Issue 4, 2018, Pages 308-316, ISSN 2213-1779.
4) *Maja Cikes, Sergio Sanchez-Martinez, Brian Claggett, Nicolas Duchateau, Gemma Piella, Constantine Butakoff, Anne Catherine Pouleur, Dorit Knappe, Tor Biering-Sørensen, Valentina Kutyifa, Arthur Moss, Kenneth Stein, Scott D. Solomon3 and Bart*

*Bijnen*s. Machine learning-based phenogrouping in heart failure to identify responders to cardiac resynchronization therapy. European Journal of Heart Failure (2019) **21**, 74–85

5) *Saqib Ejaz Awan, Ferdous Sohel, Frank Mario Sanfilippo , Mohammed Bennamoun, Girish Dwivedi*. Machine learning in heart failure: ready for prime time. Curr Opin Cardiol. 2018 Mar;33(2):190-195.
6) Sanchez-Martinez S, Duchateau N, Erdei T, Fraser AG, Bijnens BH, Piella G. Characterization of myocardial motion patterns by unsupervised multiple kernel learning. *Med Image Anal* 2017;**35**:70–8
7) Shah SJ, Katz DH, Selvaraj S, Burke MA, Yancy CW, Gheorghiade M, Bonow RO, Huang CC, Deo RC. Phenomapping for novel classification of heart failure with preserved ejection fraction. *Circulation* 2015;**131**:269–279.
8) J. L. Bruse et al., "Detecting Clinically Meaningful Shape Clusters in Medical Image Data: Metrics Analysis for Hierarchical Clustering Applied to Healthy and Pathological Aortic Arches," in IEEE Transactions on Biomedical Engineering, vol. 64, no. 10, pp. 2373-2383, Oct. 2017, doi: 10.1109/TBME.2017.2655364.
9) *Matthew W. Segar, Kershaw V. Patel, Colby Ayers, Mujeeb Basit, W.H. Wilson Tang, Duwayne Willett, Jarett Berry, Justin L. Grodin, and Ambarish Pandey*. Phenomapping of patients with heart failure with preserved ejection fraction using machine learning-based unsupervised cluster analysis. European Journal of Heart Failure (2020) **22**, 148–158. doi:10.1002/ejhf.1621
10) *Jiangang Zou, Wei Hua, Yangang Su, Geng Xu, Liangrong Zhen.* SPECT-Guided LV Lead Placement for Incremental CRT Efficacy. Validated by a Prospective, Randomized, Controlled Study.
11) Peix, A., Karthikeyan, G., Massardo, T. *et al.* Value of intraventricular dyssynchrony assessment by gated-SPECT myocardial perfusion imaging in the management of heart failure patients undergoing cardiac resynchronization therapy (VISION-CRT). *J. Nucl. Cardiol.* (2019).
12) Cikes M, Solomon SD. Beyond ejection fraction: an integrative approach for assessment of cardiac structure and function in heart failure. *Eur Heart J* 2016;**37**:1642 – 1650.
13) Chung ES, Leon AR, Tavazzi L, Sun JP, Nihoyannopoulos P, Merlino J, Abraham WT, Ghio S, Leclercq C, Bax JJ, Yu CM, Gorcsan J, Sutton MS, De Sutter J, Murillo J. Results of the predictors of response to CRT (PROSPECT) trial. *Circulation* 2008;**117**:2608 – 2616.

APPENDIX: PYTHON SOURCE CODE

NAME: RUKAYAT ADEOSUN

TITLE: HIERARCHICAL CLUSTERING

```
# Importing libraries
import pandas as pd
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
from scipy.cluster import hierarchy
import numpy as np
```

```
# Importing the data set
df = pd.read_excel('/Users/rukayatadeosun/Downloads/data1.xlsx')
```

```
# To create a new column to indicate presence/absence of response
new_label = []
for value in df["response"]:
    if value <= 0:
        new_label.append("__NORES")
    else:
        new_label.append("__RES")

df["New_label"] = new_label
pd.set_option('display.max_columns', 500,'display.max_rows', 500)

#print(df)
```

```
# To check the columns in the dataset
df.columns
```

```
# create a new column ID (New_ID) to label HC by concatenating columns ID and New label
df['New_ID'] = df[['ID', 'New_label']].apply(lambda x: ''.join(x), axis=1)
df
```

```
# To check the new column
df['New_ID']
```

```
# To drop string labels
del df['ID']
del df['response']
del df['New_label']

# To drop some unimportant columns with missing data
del df['Echo_pre_LVEF']
del df['height']
del df['weight']
```

```
# To set the column new ID as index
df1 = df.set_index('New_ID', inplace=False)
print(df1.shape)
print(df1)
```

```
# To calculate the count and percentage of categorical variables in the data
# Iterate over several given columns
# only from the dataframe
for column in df1[['ACEI_or_ARB', 'CAD', 'Concordance', 'DM', 'Gender', 'HTN', 'LBBB', 'MI', 'NYHA', 'Race','Smoking']]:

    # Select column contents by column
    # name using [] operator
    columnSeriesObj = df1[column]
    print('Column Name : ', column)
    print('Column Count : ', columnSeriesObj.value_counts())
    print('Column Percentage : ', columnSeriesObj.value_counts(normalize = True)*100)
    print('\n')
```

```
# mean of all columns
df1.mean()
```

```
# Standard deviation of all columns
df1.std()
```

```
# convert index type to list
df1.index.tolist()
```

```
# Calculate the distance between each sample
# linkage method defines how the distance between clusters is measured
# In complete linkage, the distance between clusters is the distance between the furthest points of the clusters
# In single linkage, the distance between clusters is the distance between the closest points of the clusters.

# Perform the necessary imports
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt

# Calculate the linkage: mergings
mergings = linkage(df1, method='complete')

# Plot the dendrogram, using varieties as labels
plt.figure(figsize=(35,15))
dendrogram(mergings,
           labels=df1.index.tolist(),
           leaf_rotation=90,
           leaf_font_size=8,color_threshold=240)
plt.show()
```

```
# Control number of clusters in the plot + add horizontal line.
plt.figure(figsize=(35,15))
dendrogram(mergings,
           labels=df1.index.tolist(),
           leaf_rotation=90,
           leaf_font_size=8,color_threshold=240)
plt.axhline(y=400, c='black', lw=1, linestyle='dashed')
plt.show()
```

```
# reshuffled the dataframe to confirm accuracy of HC
df_shuf = df1.sample(frac=1)
df_shuf
```

```
# part of the reshuffling process
df_shuf1 = df1.sample(frac=1).reset_index(drop=True)
df_shuf1 = df.set_index('New_ID', inplace=False)
df_shuf1
```

```
# converted the dataframe to list to be used as labels for the HC
df_shuf.index.tolist()
```

```
# conversion of dataframe column to list to be used as labels for the HC
df_shuf1.index.tolist()
```

```
# Perform the necessary imports
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt

# Calculate the linkage: mergings
mergings1 = linkage(df_shuf, method='complete')

# Plot the dendrogram, using varieties as labels
plt.figure(figsize=(35,15))
dendrogram(mergings1,
           labels=df_shuf.index.tolist(),
           leaf_rotation=90,
           leaf_font_size=8,color_threshold=240)
plt.show()
```

```
# Control number of clusters in the plot + add horizontal line.
plt.figure(figsize=(35,15))
dendrogram(mergings1,
           labels=df_shuf.index.tolist(),
           leaf_rotation=90,
           leaf_font_size=8,color_threshold=240)
```

```
# Calculate the linkage: mergings
mergings1 = linkage(df_shuf, method='complete')

# Plot the dendrogram, using varieties as labels
plt.figure(figsize=(35,15))
dendrogram(mergings1,
           labels=df_shuf.index.tolist(),
           leaf_rotation=90,
           leaf_font_size=8,color_threshold=240)
plt.show()
```

```
# Control number of clusters in the plot + add horizontal line.
plt.figure(figsize=(35,15))
dendrogram(mergings1,
           labels=df_shuf.index.tolist(),
           leaf_rotation=90,
           leaf_font_size=8,color_threshold=240)
plt.axhline(y=400, c='black', lw=1, linestyle='dashed')
plt.show()
```

```
# Perform the necessary imports
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt

# Calculate the linkage: mergings
mergings2 = linkage(df_shuf1, method='complete')

# Plot the dendrogram, using index as labels
plt.figure(figsize=(35,15))
dendrogram(mergings2,
           labels=df_shuf1.index.tolist(),
           leaf_rotation=90,
           leaf_font_size=8,color_threshold=240)
plt.show()
```

```
# Control number of clusters in the plot + add horizontal line.
plt.figure(figsize=(35,15))
dendrogram(mergings2,
           labels=df_shuf1.index.tolist(),
           leaf_rotation=90,
           leaf_font_size=8,color_threshold=240)
plt.axhline(y=400, c='black', lw=1, linestyle='dashed')
plt.show()
```

Since shuffling the dataset gives same dendrogram, we will go ahead and analyze the clusters

```
# Perform the necessary imports
import pandas as pd
from scipy.cluster.hierarchy import fcluster

# Use fcluster to extract labels: labels
labels = fcluster(mergings, 400, criterion='distance')
print(labels)
print(labels.shape)
```

```
# Create a DataFrame with labels and Response/No_response as columns: df2
df2 = pd.DataFrame({'LABELS': labels, 'Res/No_Res': df1.index.tolist()})
arranged_labels = df2.sort_values('LABELS')

arranged_labels
```

```
# Create crosstab: ct
ct = pd.crosstab(df2['LABELS'], df2['Res/No_Res'])
# Display ct
ct
```

```
# To check cluster statistics
arranged_labels.describe()
```

```
# To check the information in the arranged labels
arranged_labels.info()
```

```
# To cull out cluster 1
lab1 = arranged_labels[(arranged_labels['LABELS']==1)]
print(lab1)
print(lab1.count())
```

```
# To filter patients with response in phenogroup 1
# String to be searched in end of string
search1 ="_NORES"
search2 ="_RES"

# boolean series returned with False at place of NaN
NORES_df1 = lab1["Res/No_Res"].str.endswith(search1, na = False)
RES_df1 = lab1["Res/No_Res"].str.endswith(search2, na = False)

# displaying filtered dataframe
print(lab1[NORES_df1])
print(lab1[NORES_df1].count())
print('\n')
print(lab1[RES_df1])
print(lab1[RES_df1].count())
```

```
# Set index for cluster 1
print(lab1.shape)
lab1 = lab1.set_index('Res/No_Res', inplace=False)
print(lab1.shape)
clus1 = lab1.index.tolist()

clus1
```

```
# To select rows in cluster 1
clus_1 = df1[df1.index.isin(clus1)]
print(clus_1.shape)
```

```
# To select rows in cluster 1
clus_1 = df1[df1.index.isin(clus1)]
print(clus_1.shape)
print(clus_1)
```

```
# Create a new column with its corresponding cluster number
clus_1['New_group'] = 1
print(clus_1.shape)
print(clus_1)
```

```
# To calculate the count and percentage of categorical variables in cluster 1
# Iterate over many given columns
# only from the dataframe
for column in clus_1[['ACEI_or_ARB', 'CAD', 'Concordance', 'DM', 'Gender', 'HTN', 'LBBB', 'MI', 'NYHA', 'Race','Smoking']]:

    # Select column contents by column
    # name using [] operator
    columnSeriesObj = clus_1[column]
    print('Column Name : ', column)
    print('Column Count : ', columnSeriesObj.value_counts())
    print('Column Percentage : ', columnSeriesObj.value_counts(normalize = True)*100)
    print('\n')
```

```
# To cull out cluster 2
lab2 = arranged_labels[(arranged_labels['LABELS']==2)]
print(lab2.count())
print(lab2.shape)
print(lab2)
```

```
# To filter patients with response in phenogroup 2
# String to be searched in end of string
search1 ="_NORES"
search2 ="_RES"
```

```
# To filter patients with response in phenogroup 2
# String to be searched in end of string
search1 ="_NORES"
search2 ="_RES"

# boolean series returned with False at place of NaN
NORES_df2 = lab2["Res/No_Res"].str.endswith(search1, na = False)
RES_df2 = lab2["Res/No_Res"].str.endswith(search2, na = False)

# displaying filtered dataframe
print(lab2[NORES_df2])
print(lab2[NORES_df2].count())
print('\n')
print(lab2[RES_df2])
print(lab2[RES_df2].count())
```

```
# Set index for cluster 2
print(lab2.shape)
lab2 = lab2.set_index("Res/No_Res", inplace=False)
print(lab2.shape)
clus2 = lab2.index.tolist()

clus2
```

```
# To select rows in cluster 2
clus_2 = df1[df1.index.isin(clus2)]
print(clus_2.shape)
print(clus_2)
```

```
# Create a new column with its corresponding cluster number
clus_2['New_group'] = 2
print(clus_2.shape)
print(clus_2)
```

```
# To calculate the count and percentage of categorical variables in cluster 2
# Iterate over many given columns
# only from the dataframe
for column in clus_2[['ACEI_or_ARB', 'CAD', 'Concordance', 'DM', 'Gender', 'HTN', 'LBBB', 'MI', 'NYHA', 'Race','Smoking']]:

    # Select column contents by column
    # name using [] operator
    columnSeriesObj = clus_2[column]
    print('Column Name : ', column)
    print('Column Count : ', columnSeriesObj.value_counts())
    print('Column Percentage : ', columnSeriesObj.value_counts(normalize = True)*100)
    print('\n')
```

```
# To cull out cluster 3
lab3 = arranged_labels[(arranged_labels['LABELS']==3)]
print(lab3)
print(lab3.count())
```

```
# To filter patients with response in phenogroup 3
# String to be searched in end of string
search1 ="_NORES"
search2 ="_RES"

# boolean series returned with False at place of NaN
NORES_df3 = lab3["Res/No_Res"].str.endswith(search1, na = False)
RES_df3 = lab3["Res/No_Res"].str.endswith(search2, na = False)

# displaying filtered dataframe
print(lab3[NORES_df3])
print(lab3[NORES_df3].count())
print('\n')
print(lab3[RES_df3])
print(lab3[RES_df3].count())
```

```
# Set index for cluster 3
print(lab3.shape)
lab3 = lab3.set_index('Res/No_Res', inplace=False)
print(lab3.shape)
clus3 = lab3.index.tolist()

clus3
```

```
# To select rows in cluster 3
clus_3 = df1[df1.index.isin(clus3)]
print(clus_3.shape)
print(clus_3)
```

```
# Create a new column with its corresponding cluster number
clus_3['New_group'] = 3
print(clus_3.shape)
print(clus_3)
```

```
# To calculate the count and percentage of categorical variables in cluster 3
# Iterate over many given columns
# only from the dataframe
for column in clus_3[['ACEI_or_ARB', 'CAD', 'Concordance', 'DM', 'Gender', 'HTN', 'LBBB', 'MI', 'NYHA', 'Race','Smoking']]:

    # Select column contents by column
    # name using [] operator
    columnSeriesObj = clus_3[column]
    print('Column Name : ', column)
    print('Column Count : ', columnSeriesObj.value_counts())
    print('Column Percentage : ', columnSeriesObj.value_counts(normalize = True)*100)
    print('\n')
```

```
# To cull out cluster 4
lab4 = arranged_labels[(arranged_labels['LABELS']==4)]
print(lab4)
print(lab4.count())
```

```
# To call out cluster 4
lab4 = arranged_labels[(arranged_labels['LABELS']==4)]
print(lab4)
print(lab4.count())
```

```
# To filter patients with response in phenogroup 4
# String to be searched in end of string
search1 ="_NORES"
search2 ="_RES"

# boolean series returned with False at place of NaN
NORES_df4 = lab4["Res/No_Res"].str.endswith(search1, na = False)
RES_df4 = lab4["Res/No_Res"].str.endswith(search2, na = False)

# displaying filtered dataframe
print(lab4[NORES_df4])
print(lab4[NORES_df4].count())
print('\n')
print(lab4[RES_df4])
print(lab4[RES_df4].count())
```

```
# Set index for cluster 4
print(lab4.shape)
lab4 = lab4.set_index("Res/No_Res", inplace=False)
print(lab4.shape)
clus4 = lab4.index.tolist()

clus4
```

```
# To select rows in cluster 4
clus_4 = df1[df1.index.isin(clus4)]
print(clus_4.shape)
print(clus_4)
```

```
# Create a new column with its corresponding cluster number
clus_4['New_group'] = 4
print(clus_4.shape)
```

```
# Create a new column with its corresponding cluster number
clus_4['New_group'] = 4
print(clus_4.shape)
print(clus_4)
```

```
# To calculate the count and percentage of categorical variables in cluster 4
# Iterate over many given columns
# only from the dataframe
for column in clus_4[['ACEI_or_ARB', 'CAD', 'Concordance', 'DM', 'Gender', 'HTN', 'LBBB', 'MI', 'NYHA', 'Race','Smoking']]:

    # Select column contents by column
    # name using [] operator
    columnSeriesObj = clus_4[column]
    print('Column Name : ', column)
    print('Column Count : ', columnSeriesObj.value_counts())
    print('Column Percentage : ', columnSeriesObj.value_counts(normalize = True)*100)
    print('\n')
```

```
# mean of columns in phenogroup 1
clus_1.mean()
```

```
# standard deviation of phenogroup 1
clus_1.std()
```

```
#mean of columns in phenogroup 2
clus_2.mean()
```

```
# standard deviation of columns in phenogroup 2
clus_2.std()
```

```
# mean of columns in phenogroup 3
clus_3.mean()
```

```
# standard deviation of columns in phenogroup 3
clus_3.std()
```

```
# mean of columns in phenogroup 4
clus_4.mean()
```

```
# standard deviation of columns in phenogroup 4
clus_4.std()
```

```
# Concatenating all phenogroups into a new dataframe
new_df1 = pd.concat([clus_1, clus_2, clus_3, clus_4])
print(new_df1)
```

```
# import packages
import statsmodels.api as sm
from statsmodels.formula.api import ols  #ordinary squares
```

```
# boxplot of columns age and new_group
new_df1.boxplot('Age', by='New_group')
```

```
# Corresponding ANOVA of the above cell
mod = ols('Age ~ New_group', data=new_df1).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
# boxplot of columns ECG_pre_QRSd and new_group
new_df1.boxplot('ECG_pre_QRSd', by='New_group')
```

```
# Corresponding ANOVA of the above cell
mod = ols('ECG_pre_QRSd ~ New_group', data=new_df1).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
# boxplot of columns ECG_pre_QRSd and new_group
new_df1.boxplot('ECG_pre_QRSd', by='New_group')
```

```
# Corresponding ANOVA of the above cell
mod = ols('ECG_pre_QRSd ~ New_group', data=new_df1).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
# boxplot of columns SPECT_pre_EDE and new_group
new_df1.boxplot('SPECT_pre_EDE', by='New_group')
```

```
# Corresponding ANOVA of the above cell
mod = ols('SPECT_pre_EDE ~ New_group', data=new_df1).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
# boxplot of columns SPECT_pre_EDSI and new_group
new_df1.boxplot("SPECT_pre_EDSI", by="New_group")
```

```
# Corresponding ANOVA of the above cell
mod = ols('SPECT_pre_EDSI ~ New_group', data=new_df1).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
# boxplot of columns SPECT_pre_EDV and new_group
new_df1.boxplot('SPECT_pre_EDV', by='New_group')
```

```
# Corresponding ANOVA of the above cell
mod = ols('SPECT_pre_EDV ~ New_group', data=new_df1).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
# boxplot of columns SPECT_pre_ESE and new_group
new_df1.boxplot('SPECT_pre_ESE', by='New_group')
```

```
# Corresponding ANOVA of the above cell
mod = ols('SPECT_pre_ESE ~ New_group', data=new_df1).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
# boxplot of columns SPECT_pre_ESSI and new_group
new_df1.boxplot('SPECT_pre_ESSI', by='New_group')
```

```
# Corresponding ANOVA of the above cell
mod = ols('SPECT_pre_ESSI ~ New_group', data=new_df1).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
# boxplot of columns SPECT_pre_ESV and new_group
new_df1.boxplot('SPECT_pre_ESV', by='New_group')
```

```
# Corresponding ANOVA of the above cell
mod = ols('SPECT_pre_ESV ~ New_group', data=new_df1).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
# boxplot of columns SPECT_pre_LVEF and new_group
new_df1.boxplot('SPECT_pre_LVEF', by='New_group')
```

```
# Corresponding ANOVA of the above cell
mod = ols('SPECT_pre_LVEF ~ New_group', data=new_df1).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
# boxplot of columns SPECT_pre_PBW and new_group
new_df1.boxplot('SPECT_pre_PBW', by='New_group')
```

```
# boxplot of columns SPECT_pre_PSD and new_group
new_df1.boxplot('SPECT_pre_PSD', by='New_group')
```

```
# Corresponding ANOVA of the above cell
mod = ols('SPECT_pre_PSD ~ New_group', data=new_df1).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
# boxplot of columns SPECT_pre_50scarand new_group
new_df1.boxplot('SPECT_pre_50scar', by='New_group')
```

```
# Corresponding ANOVA of the above cell
mod = ols('SPECT_pre_50scar ~ New_group', data=new_df1).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
# boxplot of columns ECHO_pre_EDV and new_group
new_df1.boxplot('Echo_pre_EDV', by='New_group')
```

```
# Corresponding ANOVA of the above cell
mod = ols('Echo_pre_EDV ~ New_group', data=new_df1).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
# boxplot of columns ECHO_pre_ESV and new_group
new_df1.boxplot('Echo_pre_ESV', by='New_group')
```

```
# Corresponding ANOVA of the above cell
mod = ols('Echo_pre_ESV ~ New_group', data=new_df1).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
# boxplot of columns LVEF and new_group
new_df1.boxplot('LVEF', by='New_group')
```

```
# Corresponding ANOVA of the above cell
mod = ols('LVEF ~ New_group', data=new_df1).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
# crosstab of the ACEI_ARB and new_group column
ACEI_crosstab = pd.crosstab(new_df1['ACEI_or_ARB'],
                            new_df1['New_group'],
                               margins = False)
print(ACEI_crosstab)
```

```
# required imports
from scipy.stats import chi2_contingency
```

```
# Get chi-square value , p-value, degrees of freedom, expected frequencies using the function chi2_contingency
stat, p, dof, expected = chi2_contingency(ACEI_crosstab)

# select significance value
alpha = 0.05
# Determine whether to reject or keep your null hypothesis
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print('Phenogroups are associated (reject H0)')
else:
    print('Variables are not associated(fail to reject H0)')
```

```
# crosstab of the CAD and new_group column
CAD_crosstab = pd.crosstab(new_df1['CAD'],
                            new_df1['New_group'],
                               margins = False)
print(CAD_crosstab)
```

```
# Get chi-square value , p-value, degrees of freedom, expected frequencies using the function chi2_contingency
stat, p, dof, expected = chi2_contingency(CAD_crosstab)

# select significance value
alpha = 0.05
# Determine whether to reject or keep your null hypothesis
print('significance=%.3f, p=%.3f' % (alpha, p))
```

```
# crosstab of the Concordance and new_group column
Con_crosstab = pd.crosstab(new_df1['Concordance'],
                           new_df1['New_group'],
                              margins = False)
print(Con_crosstab)
```

```
# Get chi-square value , p-value, degrees of freedom, expected frequencies using the function chi2_contingency
stat, p, dof, expected = chi2_contingency(Con_crosstab)

# select significance value
alpha = 0.05
# Determine whether to reject or keep your null hypothesis
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print('Variables are associated (reject H0)')
else:
    print('Variables are not associated(fail to reject H0)')
```

```
# crosstab of the DM and new_group column
DM_crosstab = pd.crosstab(new_df1['DM'],
                           new_df1['New_group'],
                              margins = False)
print(DM_crosstab)
```

```
# Chi square test of the above cell
stat, p, dof, expected = chi2_contingency(DM_crosstab)
```

```
# Chi square test of the above cell
stat, p, dof, expected = chi2_contingency(DM_crosstab)

# select significance value
alpha = 0.05
# Determine whether to reject or keep your null hypothesis
print('significance=%.3f, p=%.3f' % (alpha, p))
```

```
# crosstab of the Gender and new_group column
Gender_crosstab = pd.crosstab(new_df1['Gender'],
                              new_df1['New_group'],
                                 margins = False)
print(Gender_crosstab)
```

```
# Chi square test of the above cell
stat, p, dof, expected = chi2_contingency(Gender_crosstab)

# select significance value
alpha = 0.05
# Determine whether to reject or keep your null hypothesis
print('significance=%.3f, p=%.3f' % (alpha, p))
```

```
# crosstab of the HTN and new_group column
HTN_crosstab = pd.crosstab(new_df1['HTN'],
                              new_df1['New_group'],
                                 margins = False)
print(HTN_crosstab)
```

```
# Chi square test of the above cell
stat, p, dof, expected = chi2_contingency(HTN_crosstab)

# select significance value
alpha = 0.05
# Determine whether to reject or keep your null hypothesis
print('significance=%.3f, p=%.3f' % (alpha, p))
```

```
# Chi square test of the above cell
stat, p, dof, expected = chi2_contingency(HTN_crosstab)

# select significance value
alpha = 0.05
# Determine whether to reject or keep your null hypothesis
print('significance=%.3f, p=%.3f' % (alpha, p))
```

```
# crosstab of the LBBB and new_group column
LBBB_crosstab = pd.crosstab(new_df1['LBBB'],
                            new_df1['New_group'],
                               margins = False)
print(LBBB_crosstab)
```

```
# Chi square test of the above cell
stat, p, dof, expected = chi2_contingency(LBBB_crosstab)

# select significance value
alpha = 0.05
# Determine whether to reject or keep your null hypothesis
print('significance=%.3f, p=%.3f' % (alpha, p))
```

```
# crosstab of the MI and new_group column
MI_crosstab = pd.crosstab(new_df1['MI'],
                            new_df1['New_group'],
                               margins = False)
print(MI_crosstab)
```

```
# Chi square test of the above cell
stat, p, dof, expected = chi2_contingency(MI_crosstab)

# select significance value
alpha = 0.05
# Determine whether to reject or keep your null hypothesis
print('significance=%.3f, p=%.3f' % (alpha, p))
```

```
# crosstab of the NYHA and new_group column
NYHA_crosstab = pd.crosstab(new_df1['NYHA'],
                            new_df1['New_group'],
                               margins = False)
print(NYHA_crosstab)
```

```
# Chi square test of the above cell
stat, p, dof, expected = chi2_contingency(NYHA_crosstab)

# select significance value
alpha = 0.05
# Determine whether to reject or keep your null hypothesis
print('significance=%.3f, p=%.3f' % (alpha, p))
```

```
# crosstab of the Race and new_group column
Race_crosstab = pd.crosstab(new_df1['Race'],
                            new_df1['New_group'],
                               margins = False)
print(Race_crosstab)
```

```
# Chi square test of the above cell
stat, p, dof, expected = chi2_contingency(Race_crosstab)

# select significance value
alpha = 0.05
# Determine whether to reject or keep your null hypothesis
print('significance=%.3f, p=%.3f' % (alpha, p))
```

```
# crosstab of the Smoking and new_group column
Smoking_crosstab = pd.crosstab(new_df1['Smoking'],
                            new_df1['New_group'],
                               margins = False)
print(Smoking_crosstab)
```

```
# crosstab of the Race and new_group column
Race_crosstab = pd.crosstab(new_df1['Race'],
                            new_df1['New_group'],
                               margins = False)
print(Race_crosstab)
```

```
# Chi square test of the above cell
stat, p, dof, expected = chi2_contingency(Race_crosstab)

# select significance value
alpha = 0.05
# Determine whether to reject or keep your null hypothesis
print('significance=%.3f, p=%.3f' % (alpha, p))
```

```
# crosstab of the Smoking and new_group column
Smoking_crosstab = pd.crosstab(new_df1['Smoking'],
                            new_df1['New_group'],
                               margins = False)
print(Smoking_crosstab)
```

```
# Chi square test of the above cell
stat, p, dof, expected = chi2_contingency(Smoking_crosstab)

# select significance value
alpha = 0.05
# Determine whether to reject or keep your null hypothesis
print('significance=%.3f, p=%.3f' % (alpha, p))
```