Máster en Ingeniería de Sistemas y Control

Trabajo fin de máster

Machine learning methods to infer the metabolic and inflammatory responses to the intake of individual dietary macronutrients: beyond Polycystic Ovary Syndrome

Estudiante: **María Rodríguez Montes**

Directoras: **María Guijarro Mata-García, María Insenser Nieto**

Curso académico **2018/2019**

Convocatoria de defensa: **junio 2019**

Máster en Ingeniería de Sistemas y Control

Trabajo fin de máster

Machine learning methods to infer the metabolic and inflammatory responses to the intake of individual dietary macronutrients: beyond Polycystic Ovary Syndrome

Estudiante: **María Rodríguez Montes**

Directoras: **María Guijarro Mata-García, María Insenser Nieto**

Calificaciones

Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

We hereby authorize the Universidad Complutense and UNED to disseminate and use with academic means, non-commercial and specifically mentioning the authors, this Master's thesis report, as well as the code, documentation and/or designed prototype.

Firmado / Signed:

María
Rodríguez
Montes

Firmado digitalmente por María Rodríguez Montes
Nombre de reconocimiento (DN):
cn=María Rodríguez Montes, o=UNED, ou=Facultad de Informática, email=mrodrigue7541@alumno.uned.es, c=ES
Fecha: 2019.06.06 19:17:52 +02'00'

Firma del alumno / Student's signature

Abstract

Background: The aim of this report is to infer patterns in time series of metabolic diseases, and specifically to obtain differential patterns in women in women with polycystic ovary syndrome (PCOS), as opposed to women without PCOS and men. The influence of sex hormones on molecules related to metabolism and inflammation can contribute to to reveal the mechanisms in PCOS physiopathology. **Methods:** The methods that have been used in this analysis are centroid distance comparisons among clusters obtained by the unsupervised learning algorithm K-means. Additionally, a principal component analysis (PCA) of the whole database has been performed. Lastly, classification algorithms such as Gaussian Naive Bayes and K nearest neighbors in order to support PCOS diagnosis have been tried. **Results:** The K-means algorithm and cluster analysis concluded that differential patterns exist in variables SOP_GLP1, SOP_IL6 for women with PCOS. Additional results are given for the same analysis in women subjects, for which SOP_Pentraxin, SOG_GIP, SOL_IDL_TG, SOP_GIP show defined trends, variables SOL_IL6, SOP_Galectin, SOP_FGF21, SOG_VLDL_P, SOG_Small_VLDL_P, in men subjects, and TBARs, TG, VLDL_TG, Pentraxin when glucose is ingested, as well as GIP when protein is ingested. PCA showed that 11 components explain more than 85% of the variance of women, women with PCOS and men groups of subjects. Referring to the classification algorithms, Naive Bayes Gaussian algorithm gave 100% accuracy in the test set, while the K nearest neighbors classifier was shown not to give accurate results. **Conclusions:** This technique has been shown to have a higher contribution to patterns of trends of metabolic disorder time series, and also lower analysis times. The results have been contrasted with external research works, and the chosen features for analysis have been validated with the Gaussian Naive Bayes classifier. Focused studies for the variables obtained as results in this report can be performed in the future.

Keywords

PCOS, K-means, patterns, macronutrients, Bayes

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Polycystic ovary syndrome

Polycystic ovary syndrome (PCOS) is the most common endocrine/metabolic disorder affecting 5-10% of women of reproductive age.[1] This syndrome is considered a complex multifactorial disorder resulting from the interaction of genetic, environmental, and lifestyle influences. Women with PCOS may present with a constellation of symptoms and signs that include biochemical and/or clinical hyperandrogenism and/or ovarian dysfunction in the form of oligo-anovulation and/or polycystic ovaries.[2] PCOS is frequently associated with insulin resistance and obesity, specifically with a predominantly abdominal distribution of body fat.[3] Women with PCOS are at a higher risk of diabetes, obesity, heart disease, and high blood pressure.[4]

Postprandial dysmetabolism (abnormal metabolism of glucose and lipids after a meal) plays a major role in the pathogenesis of metabolic disorders such as obesity, diabetes and PCOS, particularly in their association with cardiovascular disease. Understanding the influence of sex hormones (such as testosterone and estradiol) on molecules related with metabolism and inflammation, particularly during the postprandial phase, may contribute to reveal the mechanisms underlying the physiopathology of PCOS. However, very little is known about the responses after a glucose, lipid or protein oral ingestion in women with PCOS compared with women and men controls. Until now, studies reported in the literature analyzed the issue from a statistical perspective[5].[6]

The aim of the present report is to determine the clinical variables with a defined pattern in women with PCOS compared with both men and control women, suggesting a specific abnormality associated with the syndrome. A series of young healthy subjects composed by women with PCOS and appropriate women and men controls were studied. The levels of a panel of metabolic and inflammatory markers have been determined after glucose, lipid and protein oral ingestion. Methods to define a specific pattern were also implemented to other two groups of patients with additional results. Moreover, a support classifier for PCOS diagnosis has been developed.

1.2 Database

The database that is used in this document is originally composed of 67 clinical variables. The clinical variables are expanded to 201 (67×3) since each of them is measured in terms of glucose load, protein load and lipid load, for which prefixes SOG, SOP and SOL are added to each variable respectively.

The majority of the analyzed variables (191 variables) were measured in three time points during the postprandial phase (time zero, 60 and 120 min after ingestion of the glucose and protein, or 120 and 240 min after the ingestion of the lipid load). 10 variables were measured in 5 time points.

Clinical variables are divided into three groups depending on the identification method: circulating, gene expression and liposcale groups. Details for each group and variable are described in sections 1.2.1, 1.2.2 and 1.2.3. The variables for each of the three groups are shown in table 1.1.

Table 1.1: Circulating, gene expression and liposcale clinical variables in the database

Type	**Clinical variables**
Molecules	COL, TG, HDL, LDL, INS, GL, TBARs, Adiponectin, Lipocalin, Adipsin, PAI, GIP, GLP1, PYY, TNF, IL6, Omentin, sLeptinR, Vaspin, Pentraxin, Galectin, FGF23, FGF21, Chemerin, Leptin
Gene expression	ADIPOR1, GHRL, GHSR, GPX1, GSR, IL10, IL10RA, IL1B, IL1R2, IL6, IL6R, LEPR, TLR2, TLR4, TNF, TNFRSF1B
Liposcale	HDL_C, HDL_P, HDL_TG, HDL_Z, IDL_C, IDL_TG, Large_HDL_P, Large_LDL_P, Large_VLDL_P, LDL_C, LDL_P, LDL_P_HDL_P, LDL_TG, LDL_Z, Medium_HDL_P, Medium_LDL_P, Medium_VLDL_P, Non_HDL_P, Small_HDL_P, Small_LDL_P, Small_VLDL_P, Total_P_HDL_P, VLDL_C, VLDL_P, VLDL_TG, VLDL_Z

An interesting note about the variables in table 1.1 is that variables TNF, IL6, IL-10, Pentraxin, Galectin and their receptors TNFRSF1B, IL6R and IL10RA have been linked, in most evidence, with chronic low-grade inflammation for PCOS patients in the fasting state.[5]

Referring to the samples, there are 53 patients in the database. For each patient, there is information about their sex and whether they have PCOS or not (women, women with PCOS and men) and whether they are obese or not. However, in this report,only the information about sex and PCOS is used, since adding the obesity variable as well as the sex and disease variable leads to groups of patients with very few members.

In the database, there are some missing values. If a time series of one variable for one patient was at least one missing value, that time series for that specific patient is dropped and not taken into account for the analysis.

A summary of the clinical experiment is provided in figure 1.1.

1.2.1 Circulating variables

The circulating group of variables consists of a set of metabolic markers with known functions in nutrient metabolism, energy homeostasis and inflammation including biochemical measurements such as glucose, insulin and cholesterol; pro-inflammatory molecules such as IL-6 and TNF-alpha, cytokines (small secreted proteins released by cells with a communication role between cells) and adipokines (substances secreted by adipocites). Multi-analyte profiling was performed in serum samples on the Luminex Magpix system (Luminex Technologies, Austin, USA) and data were analyzed with the Milliplex Analyst software version 5.1 (EDM Millipore Corporation, Massachusetts, USA).

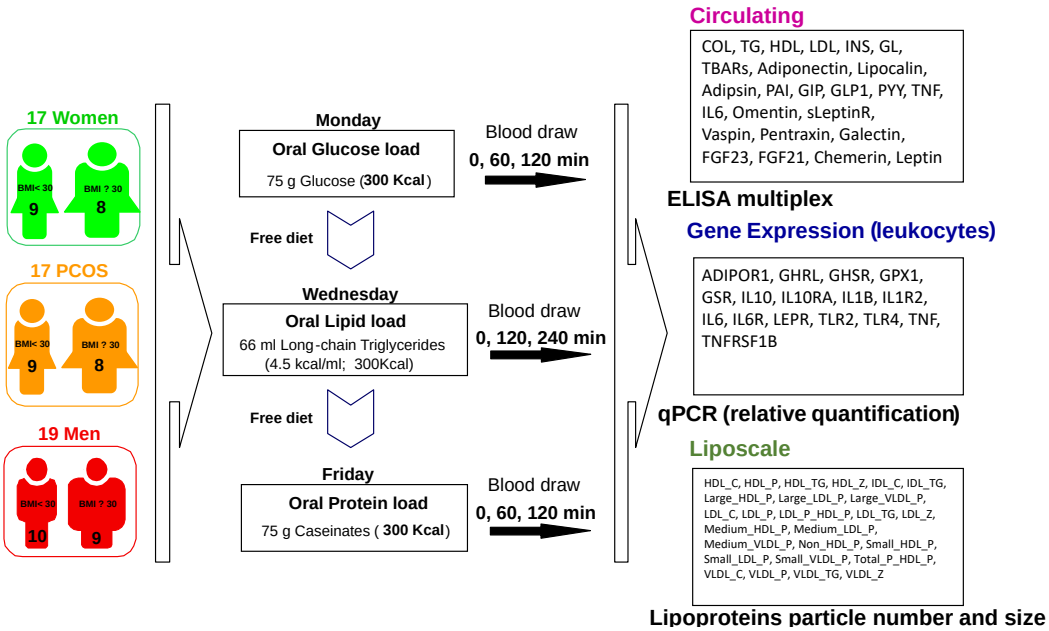

Figure 1.1: Clinical experiment scheme

Metabolic variables

GL Glucose. The main simple sugar which is an important energy source in living organisms and is a component of many carbohydrates.

COL Cholesterol. The most common type of steroid found in all the cells in the body. The body needs some cholesterol to make hormones, vitamin D, and substances that help digest foods.[4] Cholesterol is also necessary to the normal permeability and function of the membranes that surround cells.

INS Insulin. Hormone produced in the pancreas by the islets of Langerhans, which regulates the amount of glucose in the blood. The lack of insulin causes a form of diabetes.

LDL Low-density lipoprotein. Lipoprotein of blood plasma which is composed of a moderate proportion of protein with little triglyceride and a high proportion of cholesterol. It is sometimes called "bad" cholesterol because a high LDL level leads to the buildup of plaque in the arteries.

HDL High-density lipoprotein. Lipoprotein of blood plasma which is composed by a combination of a high proportion of protein with little triglyceride and cholesterol. HDL transport cholesterol from the tissues of the body to the liver, so the cholesterol can be eliminated in the bile. HDL cholesterol is considered the "good" cholesterol.

TG Triglycerides. Esters formed from glycerol and three fatty acid groups. The main constituents of natural fats and oils in the body.

FGF23	Fibroblast growth factor 23, which works as a phosphate-regulating hormone.[7]
FGF21	Fibroblast growth factor 21. Secreted endocrine factor with metabolic regulation functions such as stimulation of the uptake in adipose tissue.[8]
TBARs	Thiobarbituric Acid Reactive Substance. The TBARS assay (thiobarbituric acid reactive substance assay) is used to measure lipid peroxidation in cell and tissue extracts, and biological fluids.[9]
Inflammation variables	
TNF	Tumor necrosis factor alpha. Multifunctional proinflammatory cytokine involved in the regulation of a wide spectrum of biological processes including cell proliferation, differentiation, apoptosis, lipid metabolism, and coagulation.[10]
IL6	Interleukin 6. This is a soluble mediator with a pleiotropic (multiple) effect on inflammation, immune response, and hematopoiesis. IL-6 promptly and transiently produced in response to infections and tissue injuries, contributes to host defense through the stimulation of acute phase responses, hematopoiesis, and immune reactions.[11]
Cytokines and aditokines	
Adiponectin	Adipokine secreted by adipocytes which regulates fatty acid catabolism and glucose levels.[8]
Lipocalin	Lipocalin-2. Member of the carrier proteins superfamily. Lipocalins transport small hydrophobic molecules (e.g., steroids, bilins, retinoids and lipids) and are associated with immune response, pheromone transport, prostaglandin synthesis, retinoid binding and cancer cell interactions.[12]
Adipsin	Adiokine secreted by fat cells that improves beta cell function in diabetes.
Omentin	Adipokine that inhibits TNF-alpha-induced endothelial cell COX2 expression and induces endothelial nitric oxide synthase.[13]
Leptin	An adipokine secreted by adipose tissues with a prominent action on the hypothalamus in the brain. Leptin regulates the long-term food intake and energy expenditure.[14]
sLeptinR	Soluble leptin receptor.
Chemerin	Adipokine associated with metabolic, inflammatory, and atherosclerotic diseases.[15]

Vaspin Adipokine member of the serine protease family. Vaspin augments the plasma insulin level leading to improved glucose metabolism.[16]

Pentraxin Pentraxin-3. Multifunctional protein with complex regulatory roles in inflammation and extracellular matrix organization and remodeling.[17]

Galectin Galectin-3 is a member of a family of galactosidase-binding proteins. Galectin-3 is a potential biomarker of myocardial fibrosis in patients with heart failure.[12]

PAI Plasminogen activator inhibitor-1. Regulator of the fibrinolytic system that act by binding to and inhibiting free plasminogen activator.[12]

GIP Gastric inhibitory polypeptide. Gut hormone secreted by the mucosa of the small intestine with a role in control of gastric and intestinal secretion and insulin release.[12]

GLP1 Glucagon-like peptide 1. Gut hormone that stimulates the release of insulin from pancreatic beta cells in conjunction with carbohydrates that are absorbed from the gut.

PYY Pancreatic peptide YY. Hormone produced by the L cells of the gut. It results in decreased vagal stimulation to the gastrointestinal tract.[12]

1.2.2 Gene expression variables

Gene expression is the process by which the genetic code-the nucleotide sequence-of a gene is used to direct protein synthesis and produce the structures of the cell.[18] All variables in the following description refer to gene expression levels of peripheral blood leukocytes using quantitative real-time PCR experiments (qPCR) to evaluate the expression of the genes.

Inflammation variables

IL10 Interleukin 10. Anti-inflammatory cytokine that plays a central role in limiting host immune response to pathogens, thereby preventing damage to the host and maintaining normal tissue homeostasis.[19]

IL10RA IL10 receptor subunit alpha.

IL1B Interleukin 1 beta gene.

IL1R2 Interleukin 1 receptor type 2. The protein encoded by this gene binds interleukin alpha (IL1A), interleukin beta (IL1B), and interleukin 1 receptor, type I(IL1R1/IL1RA), and acts as a decoy receptor that inhibits the activity of its ligands.[8]

IL6 As previously described.

IL6R Interleukin 6 receptor.

TLR2	Toll-like receptor 2. Member of the TLR protein family that plays a role in the immune system.[10]
TLR4	Toll-like receptor 4. Member of the TLR protein family that plays a role in the immune system.[10]
TNF	As previously described.
TNFRSF1B	Member of the TNF-receptor superfamily.[10]

Cytokines and adipokines

ADIPOR1	Adiponectin receptor 1. This gene encodes a protein which acts as a receptor for adiponectin.[8]
GHRL	Ghrelin. Protein with a role in appetite stimulation and in energy homeostasis.
GHSR	Growth hormone secretagogue receptor. The encoded protein may play a role in energy homeostasis and regulation of body weight.[8]
GPX1	Glutathione peroxidase 1, encodes proteins that protect cells against oxidative damage.[8]
GSR	Glutathione-disulfide reductase, encodes a central enzyme of cellular antioxidant defense.[8]
LEPR	Leptin receptor. Receptor involved in the regulation of body weight.[10]

1.2.3 Liposcale variables

The liposcale test is an advanced lipoprotein analysis based on Nuclear Magnetic Resonance (NMR) spectroscopy that directly measures lipid content, number and size of lipoprotein particles.[20] Lipoproteins are soluble complexes of proteins and lipids.

Concentration

HDL_C	High density lipoprotein cholesterol level
LDL_C	Low density lipoprotein cholesterol level
VLDL_C	Very low density lipoprotein cholesterol level
IDL_C	Intermediate density lipoprotein cholesterol levels
HDL_TG	High density lipoprotein triglycerides level
LDL_TG	Low density lipoprotein triglycerides level
VLDL_TG	Very low density lipoprotein triglycerides level
IDL_TG	Intermediate density lipoprotein triglycerides level

Particle number

HDL_P	Number of high density lipoprotein particles
LDL_P	Number of low density lipoprotein particles

VLDL_P	Number of very low density lipoprotein particles
Small_HDL_P	Number of small-sized high density lipoprotein particles
Small_LDL_P	Levels of small-sized low density lipoprotein particles
Small_VLDL_P	Number of small-sized very low density lipoprotein particles
Medium_HDL_P	Number of medium-sized high density lipoprotein particles
Medium_LDL_P	Number of medium-sized low density lipoprotein particles
Medium_VLDL_P	Number of medium-sized very low density lipoprotein particles
Large_HDL_P	Number of large-sized high density lipoprotein particles
Large_LDL_P	Number of large-sized low density lipoprotein particles
Large_VLDL_P	Number of large-sized very low density lipoprotein particles
Particle size	
HDL_Z	High density lipoprotein average particle size
LDL_Z	Low density lipoprotein average particle size
VLDL_Z	Very low density lipoprotein average particle size
Ratios	
LDL_P_HDL_P	Ratio of number of low density lipoprotein particles and high density lipoprotein particles
Total_P_HDL_P	Total number of high density lipoprotein particles
Non_HDL_P	Total number of non-high density lipoprotein particles

Chapter 2

State of the art

Machine learning is a branch of artificial intelligence. It is the study of computer algorithms that improve automatically and through experience. AI is divided into supervised learning and unsupervised learning. Supervised learning consists of classification and regression algorithms. Unsupervised learning, on the other hand, can mean clustering or association algorithms. Supervised learning's goal is mapping functions whose inputs and outputs are available, whereas in unsupervised learning there are only inputs. Both methods will be used in this report.

In this state of the art review, the study of these kinds of database with the area under the curve, the use of K-means, principal component analysis, Naive Bayes and K nearest neighbors classifiers in clinical databases will be described, based on some of the most relevant clinical publications related to these topics.

2.1 Area under the curve

Up to now, the approach for analyzing response and patterns in time series similar to the ones in this database has been performed by calculating the area under the curve of each time series, obtained by subtracting the basal measurement, normalizing by time load and applying logarithmic transformation. The area under the curve is actually the integral, so it can take either positive or negative values. Methods such as the Kolmogorove Smirnov method, univariate general linear models, Mauchly's test, the Greenhouse-Geisser correction factor, the least significance test for comparisons, and the Pearson's correlation analysis were used in order to assess relationships between variables.[5]

For example, inflammatory variables such as TNF, IL6, IL-10, Pentraxin, Galectin and their receptors TNFRSF1B, IL6R and IL10RA (all present in this project's database) were analyzed when macronutrients were ingested by groups of subjects (divided by sex, exactly like in the present database) and obesity.[5] No effect of group of subject and obesity was found on Pentraxin, Galectin or IL6, TNF, IL10 and IL10RA. However, there were some relevant results on IL6R expression, which was increased in obese men and PCOS patients compared with their non-obese counterparts. Similarly, TNFRSF1B expression showed an interaction between group of subjects and obesity, consisting of obese men and women with PCOS showing increased levels when compared with their nonobese counterparts. These results imply, in part, that the clinical variables IL6R and TNFRSF1B in obese and PCOS patients are considered to have a relevant trend when compared to the rest of the groups.

In this report, the area under the curve is not the parameter that is analyzed. The goal is to analyze the trend in each time series, and the area under the curve has two problems in this regard:

- Misinformation when the values change sign in the same curve.
- Different curves can have the same area but different trends.

Therefore, it is believed that results obtained in the present document will differ other studies. The previous bullet points are proved in figure 2.1 for gene expression variable SOL_GL measured in two women with PCOS, with respect to their initial measurement. This figure shows two curves which have approximately the same integral (-1320 and -1566 respectively), while having visibly different trends in the last part of the time series. In one patient, also, the values sign in the final part of the series, and the integral does not give an accurate value of the area since some of it is canceled out.

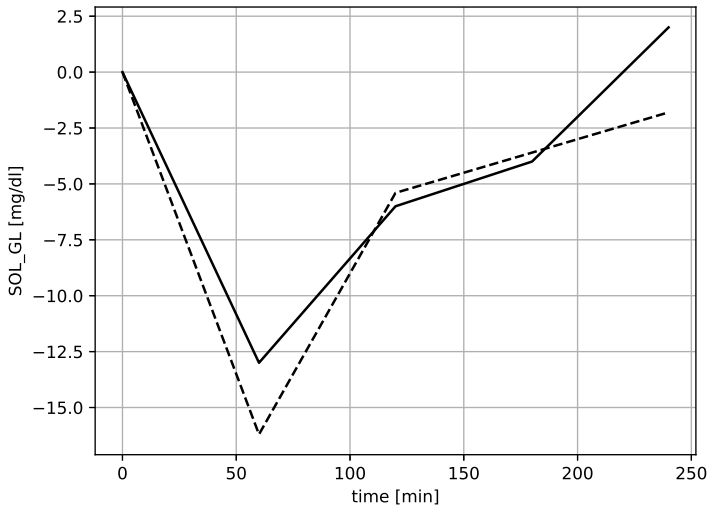

Figure 2.1: SOL_GL in two PCOS women from the database

Also, instead of statistics, machine learning algorithms have been applied, since they are the state of the art technology for data mining and finding patters, and they are being used in clinical fields.

2.2 K-means usage in clinical databases

Two examples for which K-means was used in clinical databases and that are related to this report is pattern clustering and patient diagnosis.

In some studies,[21] the relationship of different gene expression variables with normal-cancer tissue metastasis, distant metastasis and lymph-node metastasis in colorectal cancer is analyzed. K-means was used for this purpose, in order to identify which genes have correlation with each type of metastasis.

Discriminant analysis and K-means clustering are used to predict preeclampsia. Characteristics such as serum total cholesterol (TC), high density lipoprotein (HDL-C), low density lipoprotein (LDL-C) and triglycerides (TG) were measured in two groups of women at different stages: women with preeclampsia and healthy pregnant women.

The data at different stages is analyzed separately and not as a time series. The clinical markers for better accuracy prediction of preeclampsia among those parameters were identified, and then validated with K-means clustering.[22]

K-means is also used in the realm of breast cancer[23].[24] K-means clustering is performed to categorize breast cancer patients based on certain gene-expression attributes, and to identify genes that are more related with the prognosis of breast cancer.[23] The data to be analyzed is provided as tuples. K-means is also used to extract patterns from kinetic curves of malignant breast tumors.[24] The kinetic curves given by the method of dynamic contrast-enhanced magnetic resonance imaging (DCE-MRI). The clinical standard protocol is to apply a three-time-points (3TP) method to analyze patterns in these kinetic curves. Kinetic curves for each tumor are time dependent. In the 3TP method, two parameters are calculated from each curve: percentage enhancement (PE) and signal enhancement ratio (SER), and curves are classified into one of 6 categories. K-means was then used to compute the optimal number of categories, and the paper shows that using K-means and the corresponding cluster validation provides better results.

K-means can be used for digitalization of electrocardiogram (ECG) data that are still kept in paper in some regions.[25] The rate at which K-means provides the correct grouping given the features is shown to reach 99%.

2.3 Principal component analysis usage in clinical databases

Principal component analysis (PCA) was used in clinically in the fields of data dimensionality reduction, medical image processing and feature extraction.

Principal component analysis is sometimes described as a methodology to reduce the amount of irrelevant data and redundant information in medical fields.[26] The dimensionality reduction provided by PCAs is combined in this paper with the method of effective density clustering, a combination that is said to provide excellent performance and help doctors make diagnoses.

In an additional study, a dataset of more than 14000 Korean patients, with results, clinical opinions, age, sex and groups of 15 different diseases, is treated with deep neural networks by previously applying principal component analysis.[27] It is said in this paper that in deep neural networks there is a very high number of input and output neurons, which cause a very complex network structure that may produce overfitting. PCA is applied to the input features, in order to reduce the number of parameters to be trained by the neural network without much information loss.

In reference to medical imaging, PCA was used, for example, in Magnetic Resonance Imaging (MRI) technology. MRI technology demands improved resolution for accurate analysis and treatment monitoring. However, data is often affected by random noise. A new algorithm combining PCA decomposition and Wiener filtering has been developed, in order to improve upon this problem.[28]

Also, the use of fMRI (functional MRI) is the state of the art tool for early Alzheimer's disease diagnosis. However, in underdeveloped countries, fMRI is unavailable so MRI is still used. A method of feature extraction has been developed based on discrete 3D wavelet transform followed by PCA, so that the feature space is transformed into a smaller feature space with linearly uncorrelated variables in MRI data from healthy subjects.[29]

PCA decomposition has also been used to extract the most relevant features in a car-

diotocography (CTG) database containing data about the fetal state.[30] Features like the beats per minute, number of fetal movements per second, and number of uterine contractions per second are included in the PCA analysis. The PCA results are then combined with Adaptive Boosting and Support Vector Machine (SVM) in order to obtain a classifier for predicting the fetal state via CTG data.

2.4 Naive Bayes classification usage in clinical databases

Topics in which Naive Bayes has been used in clinical areas are the diagnosis of diseases and pattern classification.

It is important to note that classification tools are often presented in research papers as support tools and not as diagnosis tools. A classifier can provide help to doctors in diagnostic procedures with a certain percentage of accuracy, but for now should not be used as an isolated tool for diagnoses. For example, a Naive Bayes classifier combined with a privacy preserving tool (cryptography-related) was used to produce diagnoses of the patients' risk with high accuracy, while preserving other patients' privacy in the diagnosis phase. The tool is presented as a clinical support system.[31] A similar diagnostic support tool for rare forms of the Dengue disease is provided. Both Bayes and SVM methods are used, after selecting the most important features for the disease.[32]

Apart from diagnosis, Naive Bayes-based tools can also be used for disease management, as is the case in chronic renal failure, where some attributes for a group of patients are measured per minute, and different classification techniques are compared, including Naive Bayes.[33]

Naive Bayes classification is also used in clinical approaches as a support tool to administer certain tests to patients, meaning that it can also be a pre-diagnosis tool.[34] Naive Bayes was used to see which patients should undergo a polysomnography, a standard test to diagnose obstructive sleep apnea, based on some demographic and clinical data for each patient.

Naive Bayes can also be applied to time series, over long periods of time, for potential coronary heart disease patients. The most common patterns from a database of patients are extracted with a set of techniques called temporal abstraction, and a model based on Naive Bayes classifier is presented.[35]

2.5 K nearest neighbors usage in clinical databases

The purposes of the K nearest neighbors classifier in the clinical field are sometimes similar to what is done with the Bayes classifier. For example, a disease recommender system was developed in one study, which means a K nearest neighbors multilabel classification algorithm.[36] There is also a mono-label method for distinguishing non-epileptic and epileptic patients. This is done by analyzing time series the power spectra magnitudes of electroencephalogram waveforms.[37]

K nearest neighbors is also useful for other purposes; the scope seems to be wider than the Naive Bayes algorithm. For example, it has been used not only as a diagnosis support tool but also to estimate missing values. A weighted K nearest neighbors algorithm was used to fill in missing data from laboratory measurements in intensive care unit (ICU) patients, by using similar data from other fields.[38]

Chapter 3

Methods

3.1 K-means

K-means is an unsupervised learning algorithm[39].[40] The inputs to K-means are series of data and the number of m centroids that are desired as a result. The output is m centroids and labels for each data point; each label is the centroid to which that data point is assigned. The algorithm consists of the following steps:

1. Algorithm initialization. m centroids are initialized either from selecting them randomly from the data or randomly generated. These centroids are $[\mu_1 ... \mu_j ... \mu_m]$.
2. Once the algorithm has been initialized, iteration is performed over the following steps until convergence:
   (a) Data assignment, where each data point is assigned to the nearest centroid, by obtaining the minimum of the squared Euclidean distance. For each x_i data point, a label c_i is assigned after computing the equation for each centroid j:
   $$c_i = arg\min_j \ \|x_i - \mu_j\|^2 \tag{3.1}$$
   The previous equation means that c_i is the centroid that would minimize the equation $\|x_i - \mu_j\|^2$.
   (b) Centroid update, where centroids are recomputed. New centroids are obtained by taking the mean of all data points assigned to each centroid in the previous step. The following equation reflects this step:
   $$\mu_j = \frac{\sum_{i=1}^{m} 1\{c_i = j\} \cdot x_i}{\sum_{i=1}^{m} 1\{c_i = j\}} \tag{3.2}$$
   The $1\{c_i = j\}$ part of the previous equation means that only the centroids assigned to each specific cluster are taken into account in calculations.

The previous algorithm is usually referred to as expectation-maximization algorithm. The expectation part would be the data assignment step, where points are assigned to the nearest cluster centers, and the maximization part means setting the new cluster centers to the mean.

3.2 Trend features vector

The trend features vector was initially chosen to be the following, for variables with three and five time measurements respectively:

$$\overline{v} = [m_1, m_2] \tag{3.3}$$

$$\overline{v} = [m_1, m_2, m_3, m_4] \tag{3.4}$$

m_1 means slope of the line joining initial point and second measurement, m_2 means slope of the line joining the second and final or third measurement, etc.

However, the previous approach was discarded. The reason was that in the second step of K-means algorithm, the new centroids are computed by taking the mean of all data points assigned to each centroid. Taking the mean can cancel out features, for example, time series that have opposite signs m_2 would have the information about the slope of that part of the time series canceled out. Therefore, another approach was followed.

The trend features vector is now the following for variables which have three time measurements is the following:

$$\overline{v} = [sign(m_1), sign(m_2), |m_1| + |m_2|] \tag{3.5}$$

Both the sign and the slope value is important when analyzing patterns of behavior, and this is why they have been chosen for the features vector. By separating the sign from each part of the time series from the absolute value of the information about the slopes, the latter information is ensured not to be lost.

For variables with 5 time measurements, data have been smoothed with a logarithm and adjusted to a third degree polynomial. The features vector is:

$$\overline{v} = [sign(m_1), sign(m_2), sign(m_3), sign(m_4), |p_1| + |p_2|] \tag{3.6}$$

In the previous equation, p_1 and p_2 are the linear and quadratic coefficients of the fitted polynomial. Only the two first coefficients have been chosen in order to avoid higher distortion. It also implies less distortion than adding the slopes of each part of the curve (since it implies adding four terms), while retaining information about trend in the whole curve. Changing p_1 and p_2 coefficients shows that the linear tendency and the quadratic trend of the curve increases or decreases, so they do give information about the whole series' pattern, as is shown in an arbitrary polynomial in figures 3.1 and 3.2.

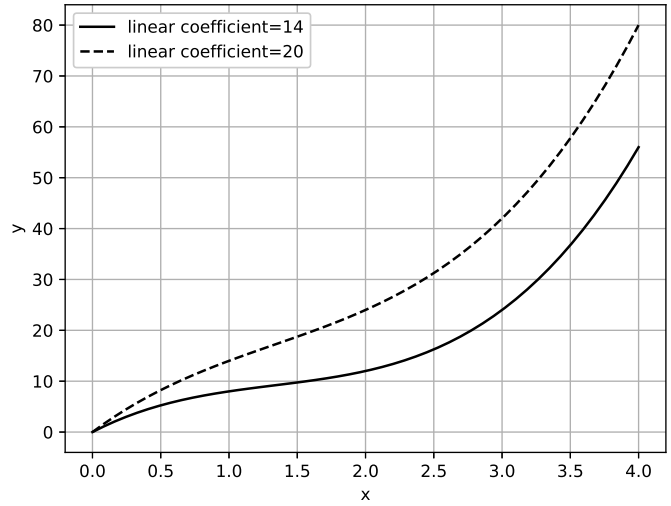

Figure 3.1: Change in linear coefficient in an arbitrary third degree polynomial

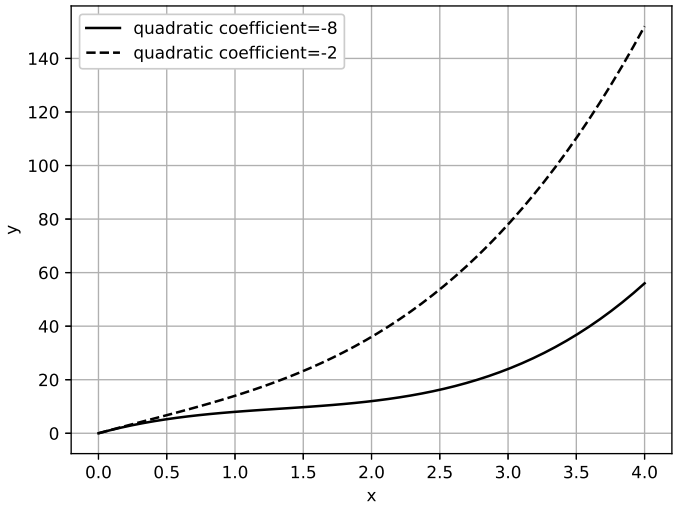

Figure 3.2: Change in quadratic coefficient in an arbitrary third degree polynomial

In some analyses in this document, the previous vectors are going to be normalized, but not in others. For example, in the K-means analysis that is going to be performed for each variable, in order to analyze distances between clusters with a unified distance for all variables, it is necessary to normalize each variable's trend vector. However, in classification algorithms, which will be described later in more detail, one feature will be taken for each clinical variable. This feature is the norm of the feature's vector, and in this case it cannot be normalized, since, if normalized, all features would have the same value and the algorithms would be useless.

3.3 Cluster analysis

The proximity between two clusters, is measured as the distance between centroids. The similarity among clusters will be measured as a function of the distance between their centroids.[41]

The distance that has been chosen to evaluate the distance between centroids is the Euclidean distance. There are other distances available for cluster analysis, like the standard Euclidean distance or the Mahalanobis distance, but the Euclidean distance has been chosen because it is easier to visualize in plotted data.

3.4 Principal component analysis

Principal component analysis (PCA) is a linear transformation method, that uses orthogonal transformation to convert a set of observations with correlated values into a set consisting of linearly uncorrelated variables. It is a technique used for dimensionality reduction in large datasets, with the aim of not losing too much information.

For the PCA to work properly, the data needs to be scaled so that each feature has unit variance.

The steps followed by the PCA algorithm are the following:[42]

- The covariance or correlation matrix are obtained from the data are computed, and eigenvectors and eigenvalues are obtained. Another approach is to perform singular vector decomposition.
- The number of features for the subspace is chosen. This is done by calculating the explained variance ratio, which is the ratio between the variance of one principal component and the total variance. Note that the total variance can be greater than one, and this is why ratios are calculated.
- Choose the eigenvectors that correspond to the largest eigenvalues. The number of eigenvectors that have to be chosen are as many as the number of features that are wanted in the new feature space.
- The projection matrix is built from the selected eigenvectors, and the original dataset is transformed with this matrix.
- The result is that the first principal component carries the highest variance for the dataset.

3.5 Gaussian Naive Bayes classifier

The Naive Bayes classifier is briefly described in this section[43].[44] The output values of any classifier are discrete, if they were continuous, the alternative supervised learning method to use would be regression.

The Bayes theorem is the following. It includes conditional probabilities ($P(A|B)$ is the probability of A given B):

$$P(c|\mathbf{X}) = \frac{P(\mathbf{X}|c)P(c)}{P(\mathbf{X})} \tag{3.7}$$

For the problem of classification with Bayes theorem, the goal is to find the value of the label c that maximizes $P(c|\mathbf{X})$. It is assumed that c is the label for each sample or observation, and $\mathbf{X}$ is the vector of n features for each sample: $\mathbf{X} = \{x_1, x_2, ...x_n\}$:

$$c = arg\max_{c \in C} P(c|\mathbf{X}) = arg\max_{c \in C} \frac{P(\mathbf{X}|c)P(c)}{P(\mathbf{X})} \tag{3.8}$$

Dropping the denominator from the previous equation:

$$c = arg\max_{c \in C} P(\mathbf{X}|c)P(c) = arg\max_{c \in C} P(\{x_1, x_2, ...x_n\}|c)P(c) \tag{3.9}$$

The Naive assumption is the each feature of $\mathbf{X}$ is independent of the others given an output or label. This assumption is rarely true, but it is useful because it allows us to make predictions on huge feature spaces, such as the one dealt with in this document. The following equation includes the Naive assumption:

$$c = arg\max_{c \in C} P(x_1|c) \cdot P(x_2|c) \cdot ...P(x_n|c) = arg\max_{c \in C} \prod_{x \in \mathbf{X}} P(x|c) \tag{3.10}$$

The Gaussian assumption means that since the features are real-valued and continuous, it is assumed that each of the probability distributions in the previous equation are normal distributions (Gaussian).

3.6 K nearest neighbors classifier

The K nearest neighbors classifier is another classifier for training labeled data and testing unlabeled data.[45] The features that are input to this classifier should be on the same scale, and the number of nearest neighbors K should be set at the start. Choosing K is often data dependent, and often a K either too small or too big is not ideal.

This classification procedure is non-parametric, which the data is not assumed to be distributed in any way. For instance, a Gaussian distribution would not have to be assumed for the data if it is not.

The procedure is also instance-based, so it does not explicitly learn a model. Only when a query is made will the algorithm use the training instances to give an answer.

Given an observation x, a distance metric d and a number of nearest neighbors K, the algorithm performs the following steps:

1. The algorithm goes through the data computing the distance d between the new observation x and each training sample. The algorithm then chooses the K training points closest to x, forming set A. For this reason the number of nearest neighbors should best be set to an even number, in order to avoid ties.
2. The algorithm estimates the conditional probability for each class, and the input gets assigned to the class c with the largest probability. This is expressed as the fraction of points of set A with that given class label:

$$c = arg\max_{c \in C} P(c|x) = arg\max_{c \in C} \frac{1}{K} \sum_{i \in A} 1\{c_i = c\} \tag{3.11}$$

3.7 Software

The software used for the analysis in this report is Python 3.7. The library that is used for the machine learning parts of this project is scikit-learn, which currently has the following capabilities: classification, regression, clustering, dimensionality reduction, model selection and preprocessing.[46] Note that the default metric for the K nearest neighbors algorithm, which is going to be used later in this document, in scikit-learn, is the Minkowski distance.

Chapter 4

Results

4.1 Variables with defined pattern in PCOS class

For each clinical variable of the database, patients have been clustered into 3 groups with the K-means algorithm. The clustering has been performed either in 3D or 5D, given the normalized trend features vector defined in previous sections. The vector has been normalized since the same criterion for Euclidean distance is going to be used for comparing centroids in all clinical variables.

$$\overline{v}_{3pts} = \frac{[sign(m_1), sign(m_2), |m_1| + |m_2|]}{\|[sign(m_1), sign(m_2), |m_1| + |m_2|]\|_{l^2}} \tag{4.1}$$

$$\overline{v}_{5pts} = \frac{[sign(m_1), sign(m_2), sign(m_3), sign(m_4), |p_1| + |p_2|]}{\|[sign(m_1), sign(m_2), sign(m_3), sign(m_4), |p_1| + |p_2|]\|_{l^2}} \tag{4.2}$$

In the previous equations, l^2 is the Euclidean norm.

The resulting centroids from K-means have been compared with the centroids of the true groups (women, women with PCOS and men) for these same features.

To determine the variables that have a defined pattern of behavior for PCOS patients, and whose features are different from those of women and men, the following conditions have to be checked:

- The true PCOS centroid has to be closer than $d = 0.3$ (Euclidean) from any K-means centroid. A distance of 0.3 has been selected since output percentage of variables with a defined pattern is considered extensive enough. A Euclidean distance of 0.1 provided 4 out of 201 variables with a defined pattern (about 0.5%), a distance of 0.2 provided a percentage of 2%, and a distance of 0.3 gave out a percentage of 5.5% variables with defined patterns in the sex groups.
- The true healthy women and men centroids are further than $d = 0.3$ (Euclidean) from that same (K-means) centroid selected in the previous bullet point.

If a clinical variable passes the two mentioned conditions, it is appended to the list of clinical variables that have a homogeneous pattern for women with PCOS. The procedure can be also applied to the women and men classes to define another two lists, or to any other group.

The final results are that variables SOP_GLP1 and circulating SOP_IL6 have a defined pattern for PCOS, and that this pattern is different from that of healthy women and men.

Centroids positions for K-means clusters and actual database centroids are shown in figures 4.1 and 4.3, as well as data points assigned to each K-means cluster by color. The original time series data is shown in figures 4.2 and 4.4. In the plots, it is somehow visible that the majority of PCOS have a more unified trend than women and men subjects. However, a mathematical algorithm is necessary to verify trend homogeneity, since patterns are not that clearly told apart at first sight.

Other studies were checked in order to validate these results' coherence. IL-6 are often elevated in patients with polycystic ovary syndrome.[47] An increase in inflammatory markers may be an early indicator of the risk of developing insulin resistance and atherosclerosis, and may become a useful prognostic and therapeutic tool for monitoring patients with PCOS.[48] This pattern in IL-6 during protein ingestion was not found in previous studies with the same database but applying statistical analysis.[5]

GLP-1 levels in response to standardized meal tolerance test were significantly reduced in women with PCOS compared to controls.[49] The effect of protein ingestion in GLP-1 secretion has not yet been described. Disturbance in gut hormones secretion dynamics might contribute to the risk of impaired glucose tolerance and type 2 diabetes in PCOS.

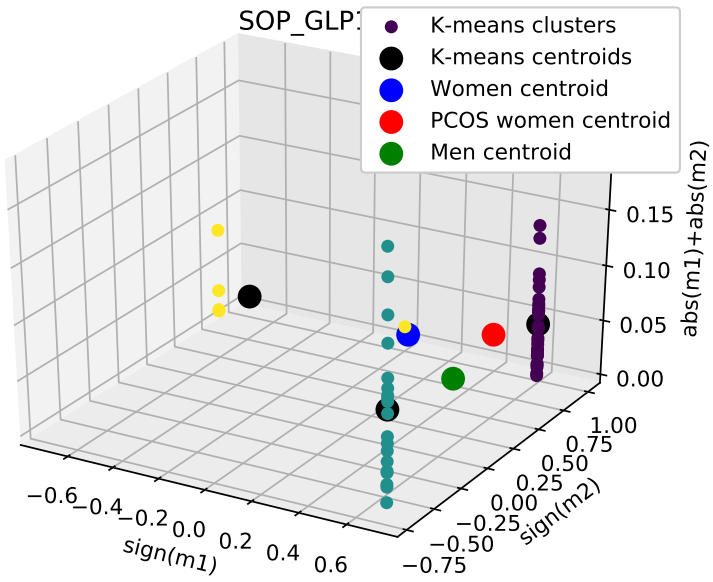

Figure 4.1: K-means algorithm results and group centroids for SOP_GLP1

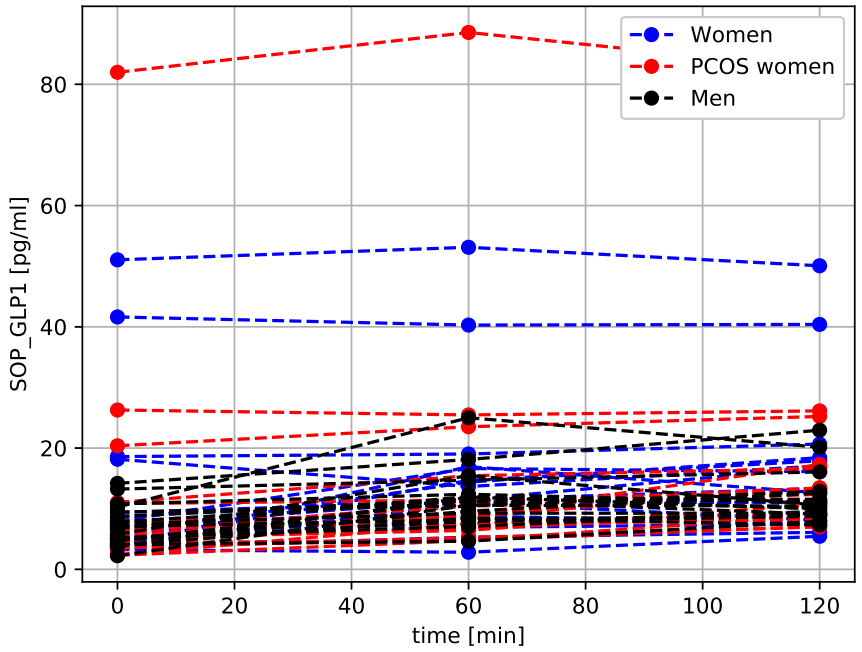

Figure 4.2: Data as a function of time for women, women with PCOS and men, SOP_GLP1

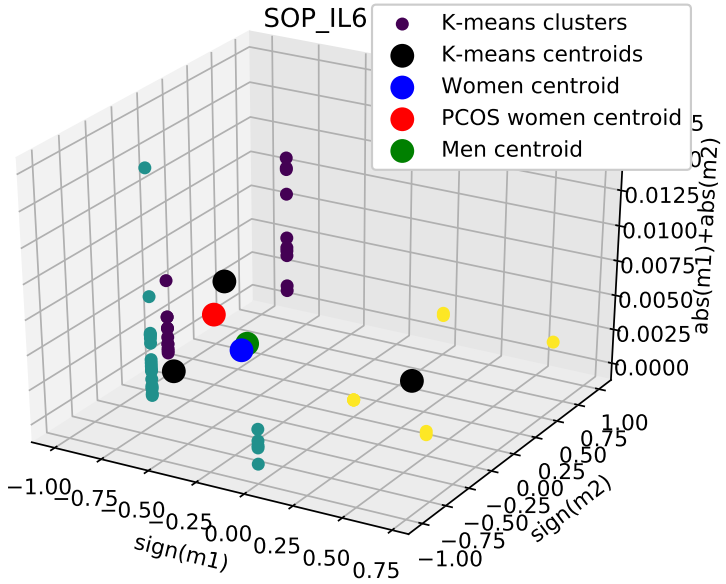

Figure 4.3: K-means algorithm results and group centroids for SOP_IL6

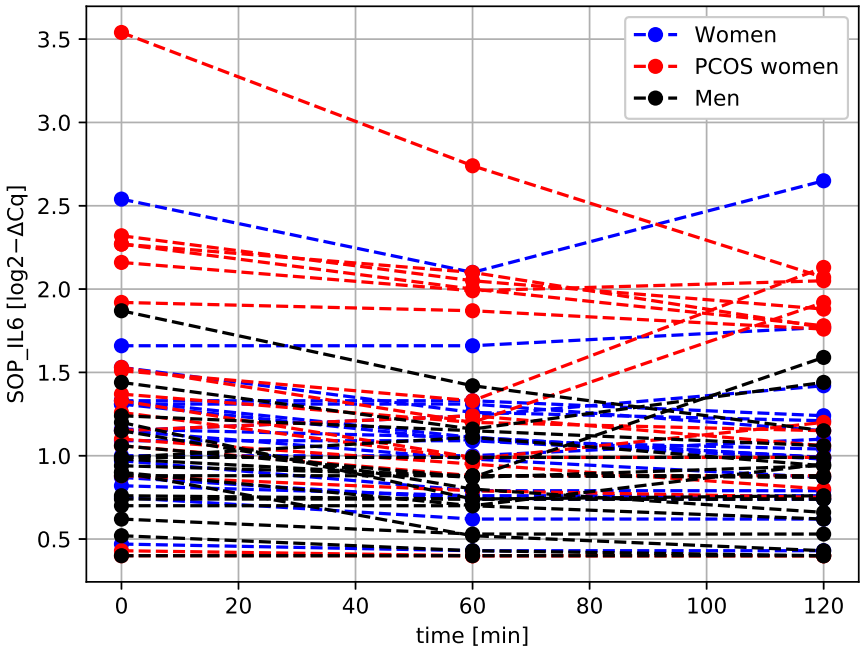

Figure 4.4: Data as a function of time for women, women with PCOS and men, SOP_IL6

4.2 Beyond PCOS: variables with defined patterns in women and men classes, and defined by macronutrient ingestion

Additional results are the group of variables with a defined pattern for women and men, as it has been done with PCOS women.

The same method can also be applied to variables of the glucose, lipids and protein loads classes, if we reformulate the database from sex groups to macronutrients groups. This way, the database would now have 53×3 samples and 67 clinical variables. The macronutrient that has been ingested by each sample is therefore not previously known, and has to be mapped by K-means.

A summary of the results is shown in table 4.1. Note that no variables with a defined trend have been identified for lipid load.

Table 4.1: Variables with a defined pattern per group

Group	**Clinical variables**
Women	SOP_Pentraxin, SOG_GIP, SOL_IDL_TG, SOP_GIP
PCOS	SOP_GLP1, SOP_IL6
Men	SOL_IL6, SOP_Galectin, SOP_FGF21, SOG_VLDL_P, SOG_Small_VLDL_P
SOG	TBARs, TG, VLDL_TG, Pentraxin
SOL	-
SOP	GIP

Although only 2 differences in patterns have been found in PCOS patients (SOP_GLP-1

and SOP_IL-6), a greater number of clinical variables have been found with differential patterns in women and men class and in the glucose and protein ingestion (15 more variables).

Sexual dimorphism patterns where observed in the pattern of 5 variables (SOL_IL6, SOP_Galectin, SOP_FGF21, SOG_VLDL_P, SOG_Small_VLDL_P) with different patterns in men than in PCOS women and control women. Men and women have many differing biological and physiological characteristics. Thus, it is no surprise that the control of metabolic processes and the mechanisms underlying metabolic-related diseases have sex-specific components. The study of sexual dimorphism is very topical[5051].[52] This analysis may serve as a guide for the targeted search of clinically relevant markers for sexual dimorphism and its association with metabolic dysfunction in larger series.

Glucose ingestion induced a defined pattern of TBARs, TG, VLDL_TG and Pentraxin levels that was not observed after the lipid and protein ingestion. These observations have been previously described[53].[6]

In adition, protein ingestion induced a defined pattern of GIP levels that was not observed after the lipid and glucose ingestion. These observations have been previously described in glucose and lipid ingestion but not with protein ingestion[54].[55] These results suggest that the postprandial response to protein ingestion differs from that of glucose and lipids, and that distinct mechanisms may be responsible of these changes.

A particularly visually clear example of defined patterns is shown in SOG_GIP for the women group. The trend features for the women group are similar among these subjects, and clearly different from PCOS and men subjects. The same plots as in the previous section are shown below, in figures 4.5 and 4.6. For the rest of the variables shown in table 4.1, the rationale is the same, and the corresponding plots are shown in the Appendix.

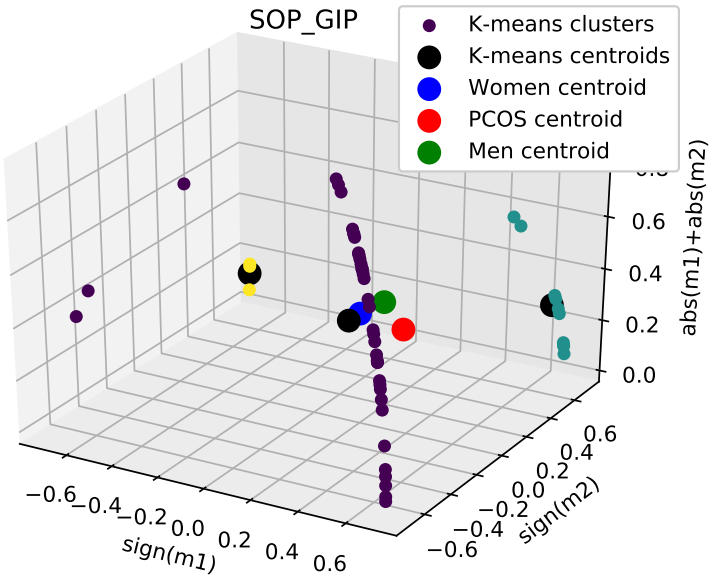

Figure 4.5: K-means algorithm results and group centroids for SOP_GIP

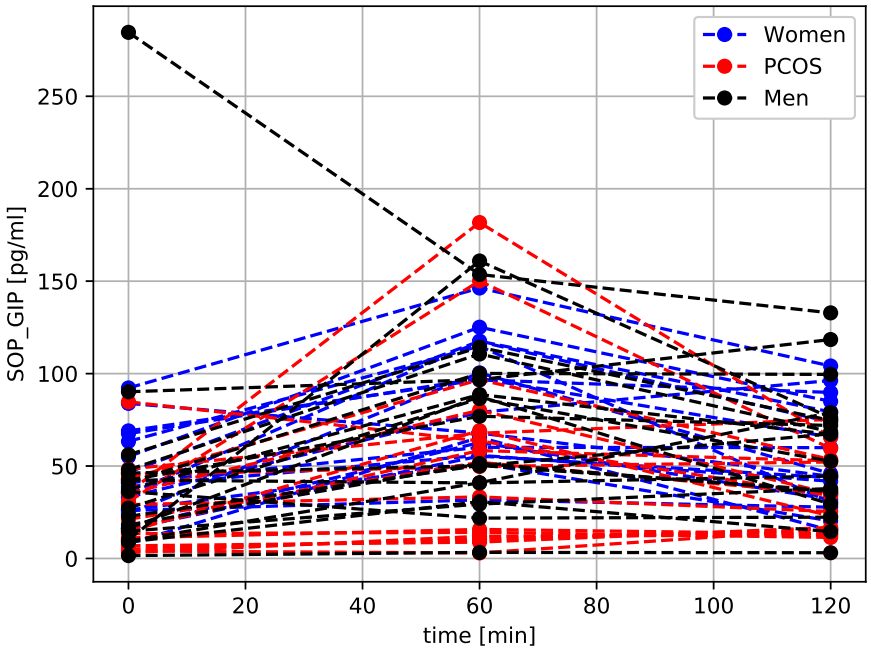

Figure 4.6: Data as a function of time for women, women with PCOS and men, SOP_GIP

4.3 Principal component analysis for women, women with PCOS and men subjects

A principal components analysis has been performed, for women, women with PCOS and men classes separately.

The studied features in this case are the norm of the features vector. This time, the vector is not normalized, as it was in the K-means analysis. The following equations show the feature that is assigned to each clinical variable:

$$feature_{3pts} = \|[sign(m_1), sign(m_2), |m_1| + |m_2|]\|_{l^2} \tag{4.3}$$

$$feature_{5pts} = \|[sign(m_1), sign(m_2), sign(m_3), sign(m_4), |p_1| + |p_2|]\|_{l^2} \tag{4.4}$$

11 principal components have been selected, since with this number of components the explained variance ratio is covered by more than 80% in the each of the three groups. Figures 4.7, 4.8 and 4.9 show the cumulative explained variance ratio as a function of the number of principal components. As there figures show, the explained variance increases as a function of the number of principal components.

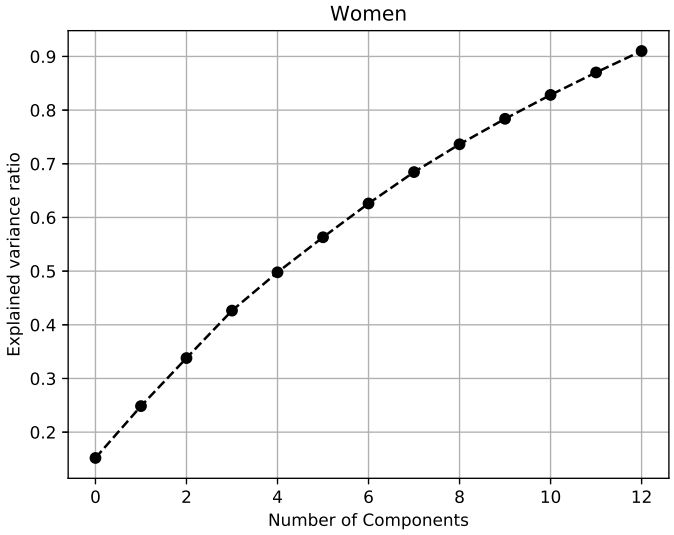

Figure 4.7: Explained variance versus number of principal components for PCA analysis in women class

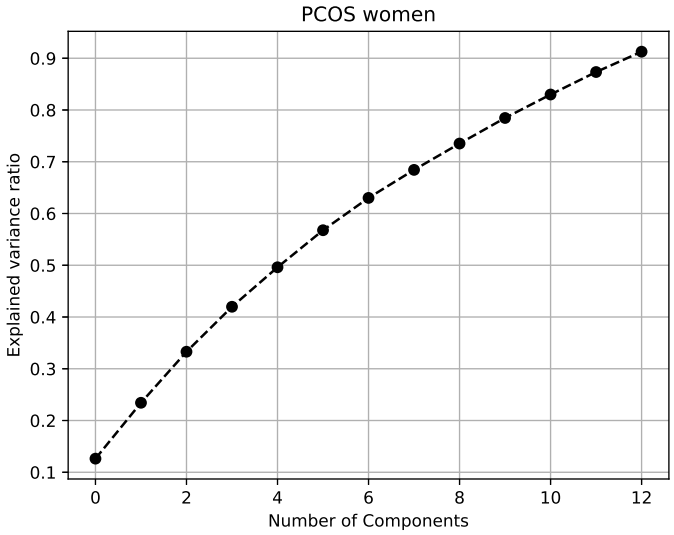

Figure 4.8: Explained variance ratio number of principal components for PCA analysis in PCOS women class

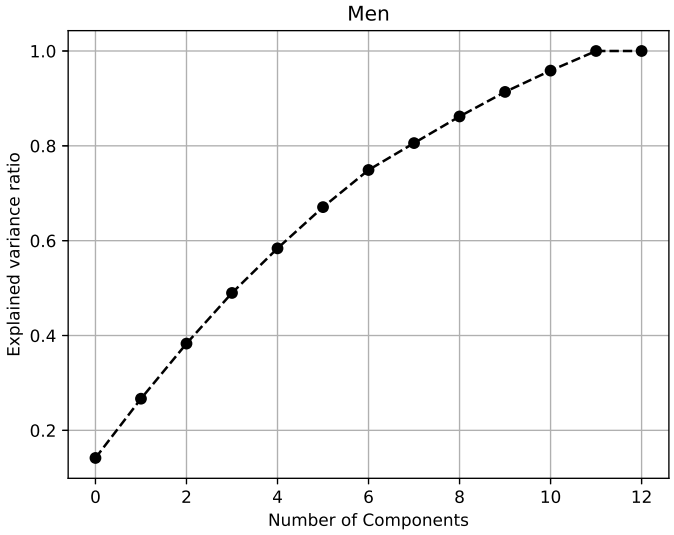

Figure 4.9: Explained variance ratio number of principal components for PCA analysis in men class

The variables that are more relevant to each of the 11 principal components for each class are shown in table 4.2.

It is observed that these variables are not consistent with those obtained in the K-means analysis (table 4.1). The only variable that coincides with 4.1 is SOG_GIP in the women group, and since it is only one clinical variable, the results are considered to be generally dissimilar.

This does not mean that the PCA results are not valid. The reason is that a different features vector is looked at to the one in the previous K-means analysis (not normalized). Also, the principal components algorithm performs an analysis on the clinical variables and their features as a whole, and not per individual clinical variable.

Table 4.2: Most relevant variables in each principal component

Group	**Clinical variables**
Women	SOL_LDL_P_HDL_P, SOG_Lipocalin, SOL_Large_LDL_P, SOG_Small_HDL_P, SOP_GL, SOG_COL, SOP_LDL_TG, SOG_Chemerin, SOG_GIP, SOP_FGF21, SOP_sLeptinR
PCOS	SOG_HDL_Z, SOL_IL1R2, SOP_Medium_LDL_P, SOP_Large_LDL_P, SOG_LDL_Z, SOP_FGF23, SOP_Pentraxin, SOL_VLDL_Z, SOP_TLR4, SOL_Small_LDL_P, SOP_TLR2
Men	SOG_LDL_P_HDL_P, SOG_TNF_ge, SOP_LDL_C, SOG_Omentin, SOG_FGF23, SOG_GHRL, SOL_TBARs, SOP_IL6, SOL_TNFRSF1B, SOP_Adiponectin, SOL_LEPR

4.4 PCOS diagnosis with classifiers

A good result for classification among women, women with PCOS and men groups has been obtained with the Gaussian Naive Bayes classifier. The same features as the ones in equations 4.3 and 4.4 are input to the classifier.

The training set consists of 70% of the data, while the rest of the samples make up the test set. A fixed random state has been taken so that the training set is always the same.

The accuracy of the Bayes classifier, in this case, is 100%. The code gives out the following results, where the test set vector is made up of 0 (women class), 1 (women with PCOS class) and 2 (men class). A new patient is also classified. This new patient is a very slight variation of a men class sample, and it is observed that the Bayes classification method likely predicts the class correctly:

```
Test set results
[1 1 1 0 1 0 2 1 1 0 1 1 1 2 0]
Classifier performance: 100.00%
New patient results
[2]
```

4.5 Negative results

In reference to PCOS diagnosis with classifiers, another classifier has been tried, K nearest neighbors, for the same features that were used in the Gaussian Naive Bayes one previously (4.3 and 4.4), and the same training and test sets.

The obtained performance for this classifier is 53.33%, which is not considered a good result. Therefore, this classifier cannot be used for PCOS diagnosis, at least when using the features described in this chapter or the variables from this database. The code output when using the K nearest neighbors classifier is the following:

```
Test set results
[1 0 0 2 0 0 2 0 1 0 1 0 0 2 0]
Classifier performance: 53.33%
New patient results
[1]
```

Chapter 5

Conclusions

This project has contributed to the study of patterns of behavior in metabolic disorder. It contributes more than performing an analysis based on the area under the curve, since there is less information loss. In addition, using machine learning techniques such as PCA and classification algorithms allows for taking into account all clinical variables at the same time. The difference between this and a statistical analysis is that the latter requires more analysis time; each clinical variable and group of subjects has to be looked at individually.

The technique used in this document has been proved to be a validated knowledge technique the etiology of metabolic disorders. The results obtained by the K-means cluster analysis have mostly been contrasted with observations in external works. For example, SOP_IL6 reactions in women with PCOS have been observed to be higher, as well as SOP_GLP1 levels have been shown to be lower when compared to control women and men, just as other research publications state, even if trends for IL-6 and GLP-1 after protein ingestion have not yet been described. The features used to describe the patterns of behavior in these time series were also internally validated in this report. The fact that the Gaussian Naive Bayes classifier gives a 100% accuracy rate is proof that new patients can be classified based on the chosen features.

Future developments of this project would be validating the chosen features with more classifiers, in order to obtain a majority of classification algorithms with high accuracy. Also, the low number of samples (53 patients) in this study posed a clear problem especially in classification algorithms, where the test set is somewhat small. It would be convenient to repeat the analysis with more subjects, so that the results are more reliable, and to potentially obtain more results.

The results obtained in this study can be used in the future to perform more focused studies, for example, instead of looking at all clinical variables, variables like SOP_IL6 and SOP_GLP1 among others can be studied more thoroughly, and subjected to new questions.

Bibliography

[1] H. F. Escobar-Morreale, "Polycystic ovary syndrome: Definition, aetiology, diagnosis and treatment," *Nature Reviews Endocrinology*, vol. 14, 03 2018.

[2] R. Azziz, E. Carmina, D. Dewailly, E. Diamanti-Kandarakis, H. F Escobar-Morreale, W. Futterweit, O. Janssen, R. Legro, R. Norman, A. E Taylor, and S. Witchel, "The androgen excess and pcos society criteria for the polycystic ovary syndrome: the complete task force report," *Fertility and sterility*, vol. 91, pp. 456–88, 11 2008.

[3] E. Carmina, S. Bucchieri, A. Esposito, A. Del Puente, P. Mansueto, F. Orio, G. Di Fede, and G. Rini, "Abdominal fat quantity and distribution in women with polycystic ovary syndrome and extent of its relation to insulin resistance," *The Journal of Clinical Endocrinology & Metabolism*, vol. 92, pp. 2500–2505, 07 2007.

[4] M. Plus. `https://medlineplus.gov`.

[5] M.-G. M.A., S. Moncayo, M. Insenser, R. Montes, E. Fernández-Durán, F. Álvarez Blasco, M. Luque-Ramírez, and H. F. Escobar-Morreale, "Postprandial inflammatory responses after oral glucose, lipid and protein challenges: Influence of obesity, sex and polycystic ovary syndrome," *Clinical Nutrition*, 04 2019.

[6] R. Montes-Nieto, M. Insenser, M. Murri, E. Fernández-Durán, M. Ojeda-Ojeda, M. A. Martínez-García, M. Luque-Ramírez, and H. F. Escobar-Morreale, "Plasma thiobarbituric acid reactive substances (tbars) in young adults: Obesity increases fasting levels only in men whereas glucose ingestion, and not protein or lipid intake, increases postprandial concentrations regardless of sex and obesity," *Molecular Nutrition & Food Research*, vol. 61, no. 11, p. 1700425, 2017.

[7] T. Saito and S. Fukumoto, "Fibroblast growth factor 23 (fgf23) and disorders of phosphate metabolism," *International journal of pediatric endocrinology*, vol. 2009, p. 496514, 10 2009.

[8] N. center for biotechnology information. `https://www.ncbi.nlm.nih.gov`.

[9] Abcam, "Tbars assay kit and tbars assay protocol." `https://www.abcam.com/kits/tbars-assay-kit-and-tbars-assay-protocol`.

[10] U. N. library of medicine, "Genetics home reference." `https://ghr.nlm.nih.gov/gene/`.

[11] T. Tanaka, M. Narazaki, and T. Kishimoto, "Il-6 in inflammation, immunity, and disease," *Cold Spring Harbor perspectives in biology*, vol. 6, 09 2014.

[12] "Segen's medical dictionary, collins dictionary of medicine, farlex partner medical dictionary, gale encyclopedia of medicine." `https://medical-dictionary.thefreedictionary.com/`.

[13] H. Yamawaki, J. Kuramoto, S. Kameshima, T. Usui, M. Okada, and Y. Hara, "Omentin, a novel adipocytokine inhibits tnf-induced vascular inflammation in human endothelial cells," *Biochemical and biophysical research communications*, vol. 408, pp. 339–43, 05 2011.

[14] H. H. N. from the Endocrine Society, "What is leptin?." `https://www.hormone.org/hormones-and-health/hormones/leptin`.

[15] L. K. Er, L.-A. Hsu, J.-M. J. Juang, F.-T. Chiang, M.-S. Teng, I.-S. Tzeng, S. Wu, J.-F. Lin, and Y.-L. Ko, "Circulating chemerin levels, but not the rarres2 polymorphisms, predict the long-term outcome of angiographically confirmed coronary artery disease," in *International journal of molecular sciences*, 2019.

[16] J. Eckel, *The cellular secretome and organ crosstalk*. Academic Press, an imprint of Elsevier, 2018.

[17] C. Garlanda, B. Bottazzi, E. Magrini, A. Inforzato, and A. Mantovani, "Ptx3, a humoral pattern recognition molecule, in innate immunity, tissue repair, and cancer," *Physiological Reviews*, vol. 98, pp. 623–639, 04 2018.

[18] U. o. L. Virtual Genetics Education Centre, "Gene expression and regulation." `https://www2.le.ac.uk/projects/vgec/highereducation/topics/geneexpression-regulation`.

[19] S. Subramanian Iyer and G. Cheng, "Role of interleukin 10 transcriptional regulation in inflammation and autoimmune disease," *Critical reviews in immunology*, vol. 32, pp. 23–63, 01 2012.

[20] L. Rubió, "Test liposcale." `http://www.laboratoriosrubio.com/wp-content/uploads/2019/04/InterpretLiposcale_ing.pdf`.

[21] M. A. Ma'sum, I. Wasito, and A. Nurhadiyatna, "Intelligent k-means clustering for expressed genes identification linked to malignancy of human colorectal carcinoma," in *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pp. 437–443, Sep. 2013.

[22] R. Mukherjee, C. D. Ray, C. Chakraborty, S. Dasgupta, and K. Chaudhury, "Clinical biomarker for predicting preeclampsia in women with abnormal lipid profile: Statistical pattern classification approach," in *2010 International Conference on Systems in Medicine and Biology*, pp. 397–401, Dec 2010.

[23] R. Radha and P. Rajendiran, "Using k-means clustering technique to study of breast cancer," in *2014 World Congress on Computing and Communication Technologies*, pp. 211–214, Feb 2014.

[24] S. H. Lee, J. H. Kim, K. G. Kim, J. S. Park, S. J. Park, and W. K. Moon, "Optimal clustering of kinetic patterns on malignant breast lesions: Comparison between k-means clustering and three-time-points method in dynamic contrast-enhanced mri," in *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 2089–2093, Aug 2007.

[25] G. Shi, G. Zheng, and M. Dai, "Ecg waveform data extraction from paper ecg recordings by k-means method," in *2011 Computing in Cardiology*, pp. 797–800, Sep. 2011.

[26] Z. Danmei, T. Qing, and X. Qijin, "A high-dimensional data analysis method based on pca and density clustering in clinical diagnosis," in *2018 Chinese Control And Decision Conference (CCDC)*, pp. 1912–1915, June 2018.

[27] Y. Kim, H. Kim, J. Hyeon, and H. Choi, "Clinical opinions generation from general blood test results using deep neural network with principle component analysis and regularization," in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 386–389, Feb 2017.

[28] L. Kanwal and M. U. Shahid, "Denoising of 3d magnetic resonance images using non-local pca and transform-domain filter," in *2016 19th International Multi-Topic Conference (INMIC)*, pp. 1–5, Dec 2016.

[29] A. Ayaz, M. Z. Ahmad, K. Khurshid, and A. M. Kamboh, "Mri based automated diagnosis of alzheimer's: Fusing 3d wavelet-features with clinical data," in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 1210–1213, July 2017.

[30] Y. Zhang and Z. Zhao, "Fetal state assessment based on cardiotocography parameters using pca and adaboost," in *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1–6, Oct 2017.

[31] X. Liu, R. Lu, J. Ma, L. Chen, and B. Qin, "Privacy-preserving patient-centric clinical decision support system on naïve bayesian classification," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, pp. 655–668, March 2016.

[32] S. Fathima and N. Hundewale, "Comparison of classification techniques-svm and naives bayes to predict the arboviral disease-dengue," in *2011 IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW)*, pp. 538–539, Nov 2011.

[33] A. Y. Al-Hyari, A. M. Al-Taee, and M. A. Al-Taee, "Clinical decision support system for diagnosis and management of chronic renal failure," in *2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, pp. 1–6, Dec 2013.

[34] D. Ferreira-Santos and P. P. Rodrigues, "Improving diagnosis in obstructive sleep apnea with clinical data: A bayesian network approach," in *2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS)*, pp. 612–617, June 2017.

[35] K. Orphanou, A. Dagliati, L. Sacchi, A. Stassopoulou, E. Keravnou, and R. Bellazzi, "Combining naive bayes classifiers with temporal association rules for coronary heart disease diagnosis," in *2016 IEEE International Conference on Healthcare Informatics (ICHI)*, pp. 81–92, Oct 2016.

[36] M. Huang, H. Han, H. Wang, L. Li, Y. Zhang, and U. A. Bhatti, "A clinical decision support framework for heterogeneous data sources," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, pp. 1824–1833, Nov 2018.

[37] X. Yu, C. Liu, J. Wang, J. Dai, J. Li, and F. Hou, "Electroencephalogram signal analysis based on the improved k-nearest neighbor network," in *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1492–1497, Oct 2016.

[38] O. T. Abdala and M. Saeed, "Estimation of missing values in clinical laboratory measurements of icu patients using a weighted k-nearest neighbors algorithm," in *Computers in Cardiology, 2004*, pp. 693–696, Sep. 2004.

[39] A. Trevino, "Introduction to k-means clustering." https://www.datascience.com/learn-data-science/tutorials/introduction-to-k-means-clustering-algorithm-data-science, June 2016. Oracle.

[40] C. Piech, "K means." http://stanford.edu/~cpiech/cs221/handouts/kmeans.html.

[41] U. de Valencia, "Introducción al análisis cluster." https://www.uv.es/ceaces/multivari/cluster/CLUSTER2.htm.

[42] S. Raschka, "Principal component analysis in python."

[43] W. Monroe, "Naïve Bayes." https://web.stanford.edu/class/archive/cs/cs109/cs109.1178/lectureHandouts/210-naive-bayes.pdf, Augusts 2017. CS 109 Lecture Notes, Stanford University.

[44] S. University, "Text classification and Naïve Bayes." https://web.stanford.edu/class/cs124/lec/naivebayes.pdf, Augusts 2017. CS 109 Lecture Notes, Stanford University.

[45] "A complete guide to k-nearest-neighbors with applications in python and r." https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/, July 2016.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[47] Z. Peng, Y. Sun, X. Lv, H. Zhang, C. Liu, and S. Dai, "Interleukin-6 levels in women with polycystic ovary syndrome: A systematic review and meta-analysis," *PloS one*, vol. 11, p. e0148531, 02 2016.

[48] A. Marciniak, J. Nawrocka-Rutkowska, A. Brodowska, B. Wiśniewska, and A. Starczewski, "Cardiovascular system diseases in patients with polycystic ovary syndrome – the role of inflammation process in this pathology and possibility of early diagnosis and prevention," *Annals of Agricultural and Environmental Medicine*, vol. 23, pp. 537–541, 09 2016.

[49] K. Aydin, G. Arusoglu, G. Koksal, N. Cinar, Y. Duygu, and B. Yildiz, "Fasting and post-prandial glucagon like peptide 1 and oral contraception in polycystic ovary syndrome," *Clinical Endocrinology*, vol. 81, 03 2014.

[50] D. C. Henstridge, J. Abildgaard, B. Lindegaard, and M. Febbraio, "Metabolic control and sex: A focus on inflammatory-linked mediators," *British Journal of Pharmacology*, 02 2019.

[51] L. Dearden, S. G Bouret, and S. Ozanne, "Sex and gender differences in developmental programming of metabolism," *Molecular Metabolism*, vol. 15, 04 2018.

[52] M. Varghese, C. N. Griffin, and K. Singer, "The role of sex and sex hormones in regulating obesity-induced inflammation.," *Advances in experimental medicine and biology*, vol. 1043, pp. 65–86, 2017.

[53] J. Rämö, S. M. Kaye, S. Jukarainen, L. Bogl, A. Hakkarainen, J. Lundbom, N. Lundbom, A. Rissanen, J. Kaprio, N. Matikainen, and K. Pietiläinen, "Liver fat and insulin sensitivity define metabolite profiles during a glucose tolerance test in young adult

twins," *The Journal of Clinical Endocrinology & Metabolism*, vol. 102, pp. jc.2015–3512, 11 2016.

[54] M. E. Kiec-Klimczak, M. Malczewska-Malec, U. Raźny, A. Zdzienicka, A. N. Gruca, J. Góralska, D. M. Pach, A. Gilis-Januszewska, A. Dembińska-Kiéc, and A. Hubalewska-Dydejczyk, "Assessment of incretins in oral glucose and lipid tolerance tests may be indicative in the diagnosis of metabolic syndrome aggravation.," *Journal of physiology and pharmacology : an official journal of the Polish Physiological Society*, vol. 67 2, pp. 217–26, 2016.

[55] S. Runchey, L. Valsta, Y. Schwarz, C. Wang, X. Song, J. W Lampe, and M. L Neuhouser, "Effect of low- and high-glycemic load on circulating incretins in a randomized clinical trial," *Metabolism: clinical and experimental*, vol. 62, 09 2012.

List of symbols

AI	Artificial intelligence
C_q	Quantification cycle
l^2	Euclidean norm
MDA	Malondialdehyde
P	Probability
PCOS	Polycystic ovary syndrome
PCR	Polymerase chain reaction
SOG	Glucose load
SOL	Lipids load
SOP	Protein load
U	Units (of insulin)

Appendix

Additional results from K-means analysis

Variables with defined pattern in women class

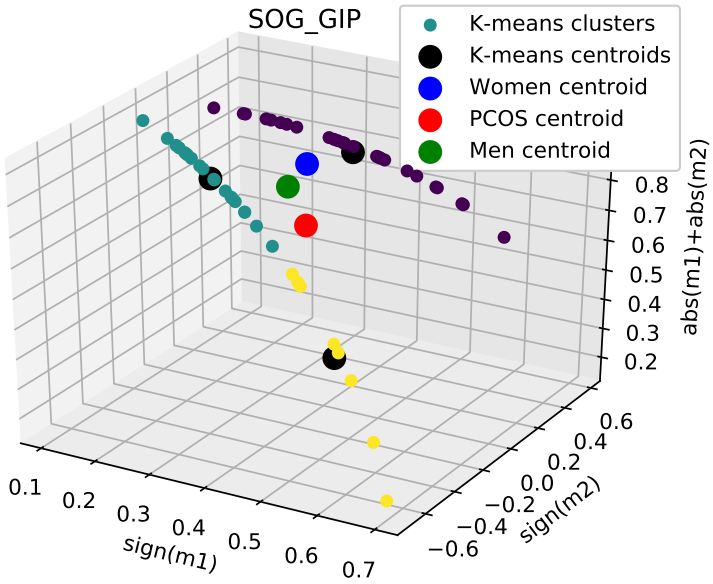

Figure 5.1: K-means algorithm results and group centroids for SOG_GIP

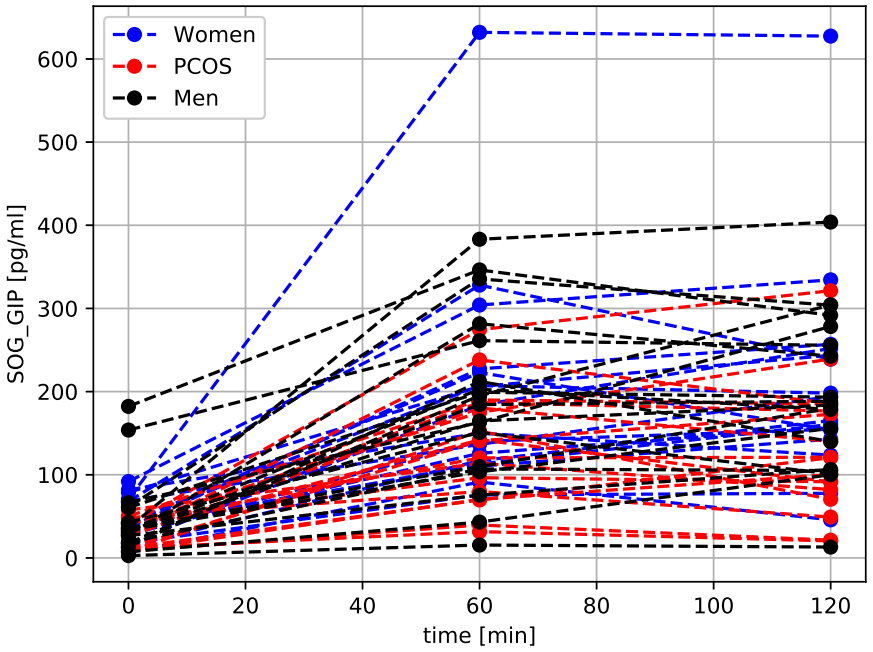

Figure 5.2: Data as a function of time for all patients, SOG_GIP

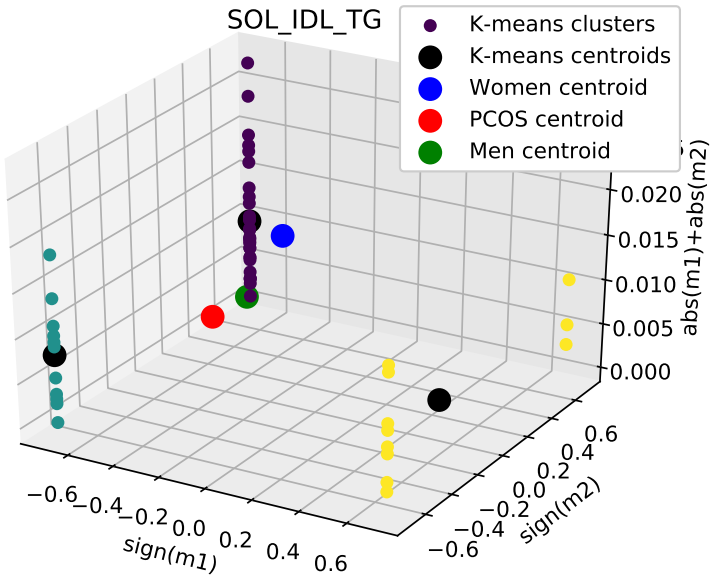

Figure 5.3: K-means algorithm results and group centroids for SOL_IDL_TG

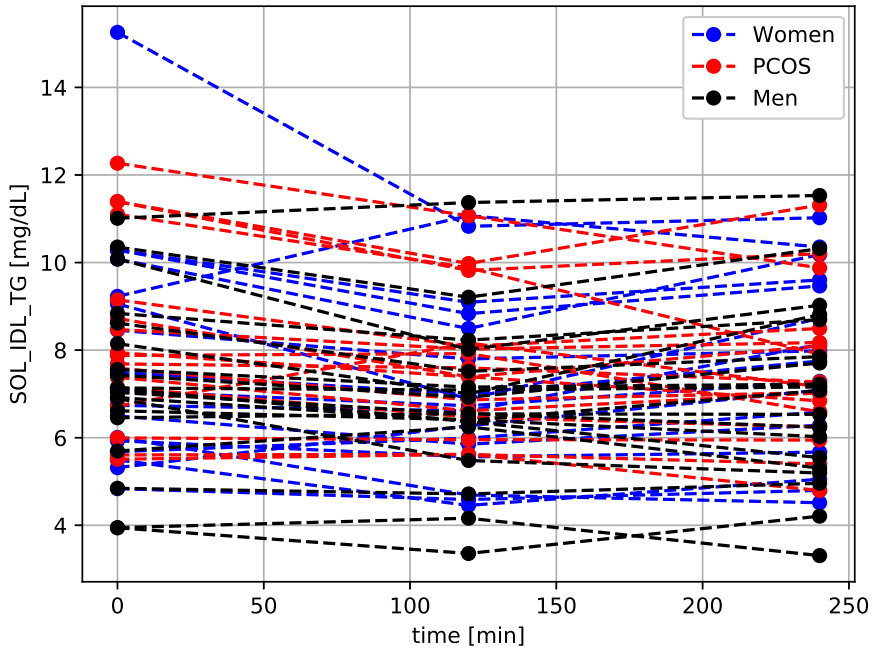

Figure 5.4: Data as a function of time for all patients, SOL_IDL_TG

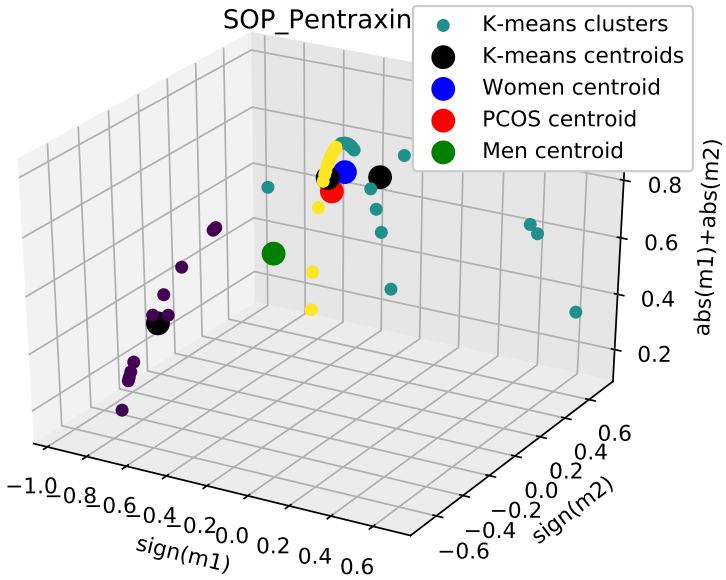

Figure 5.5: K-means algorithm results and group centroids for SOP_Pentraxin

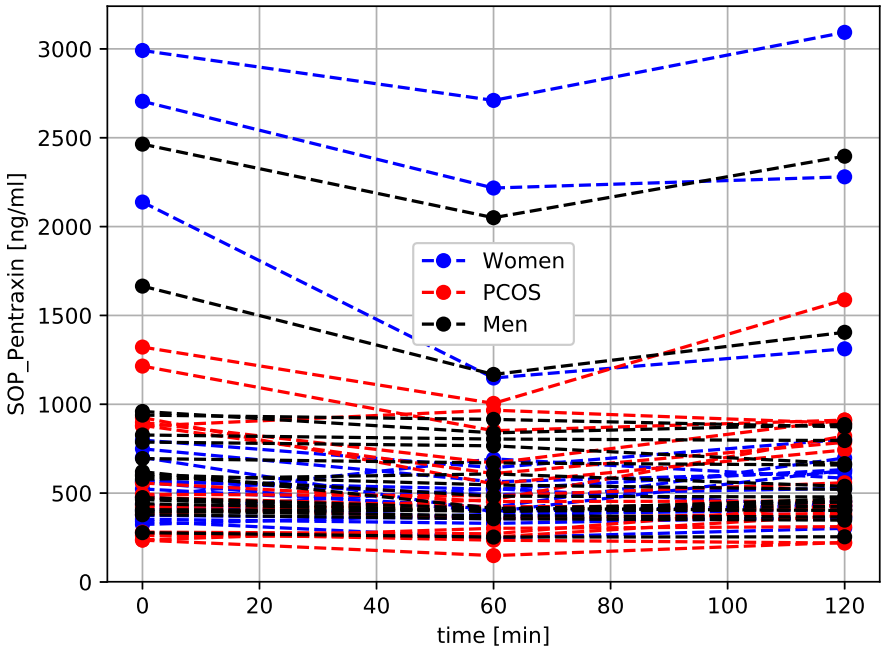

Figure 5.6: Data as a function of time for all patients, SOP_Pentraxin

Variables with defined pattern in men class

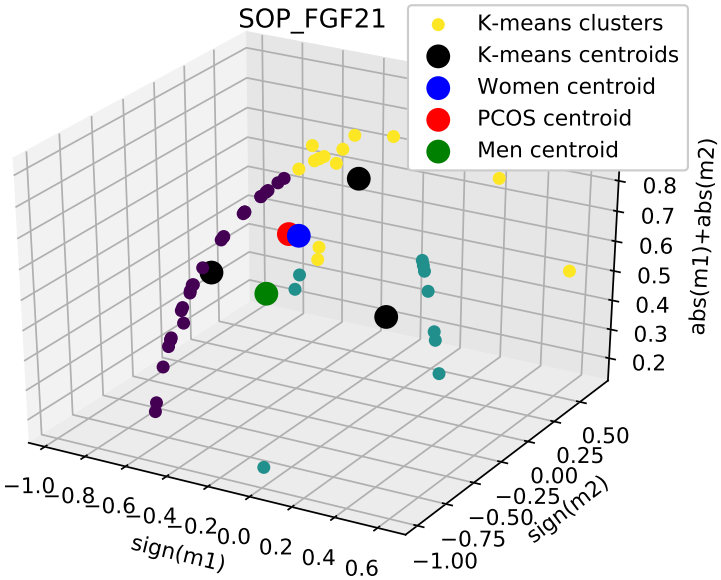

Figure 5.7: K-means algorithm results and group centroids for SOP_FGF21

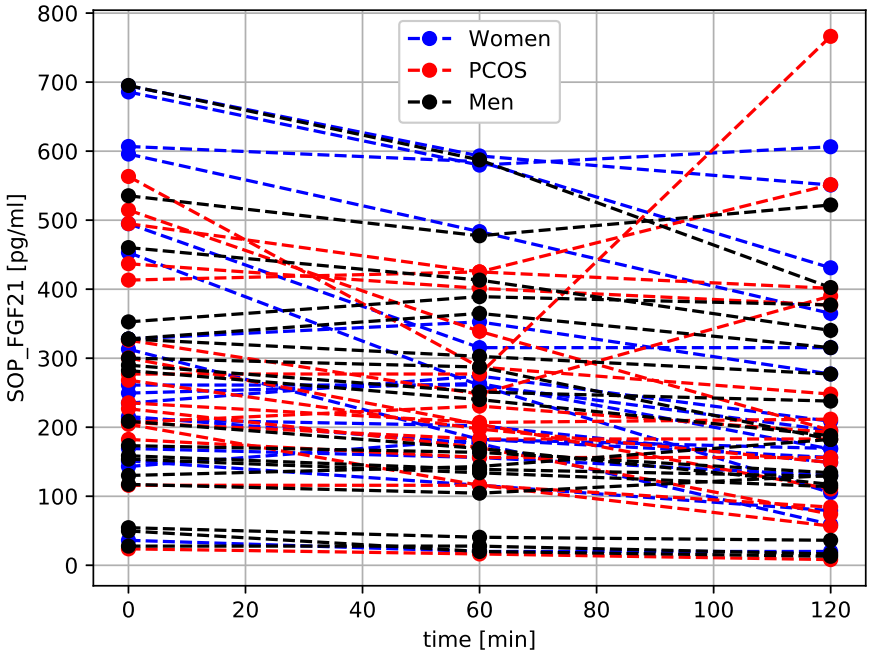

Figure 5.8: Data as a function of time for all patients, SOP_FGF21

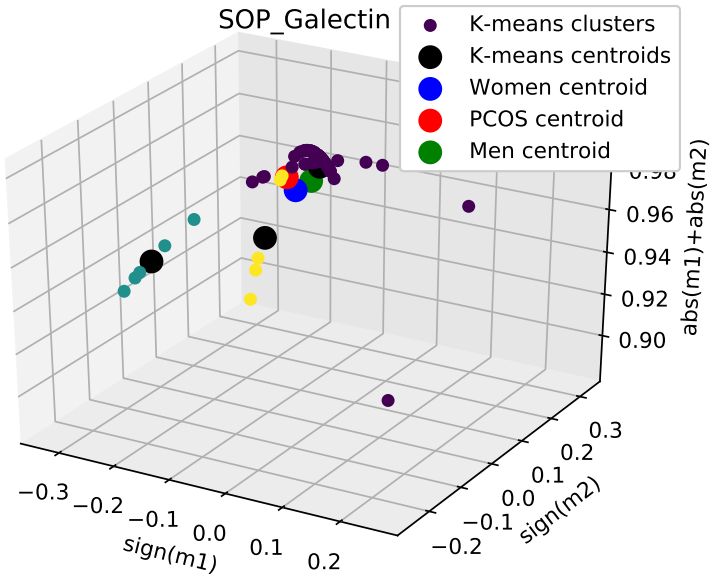

Figure 5.9: K-means algorithm results and group centroids for SOP_Galectin

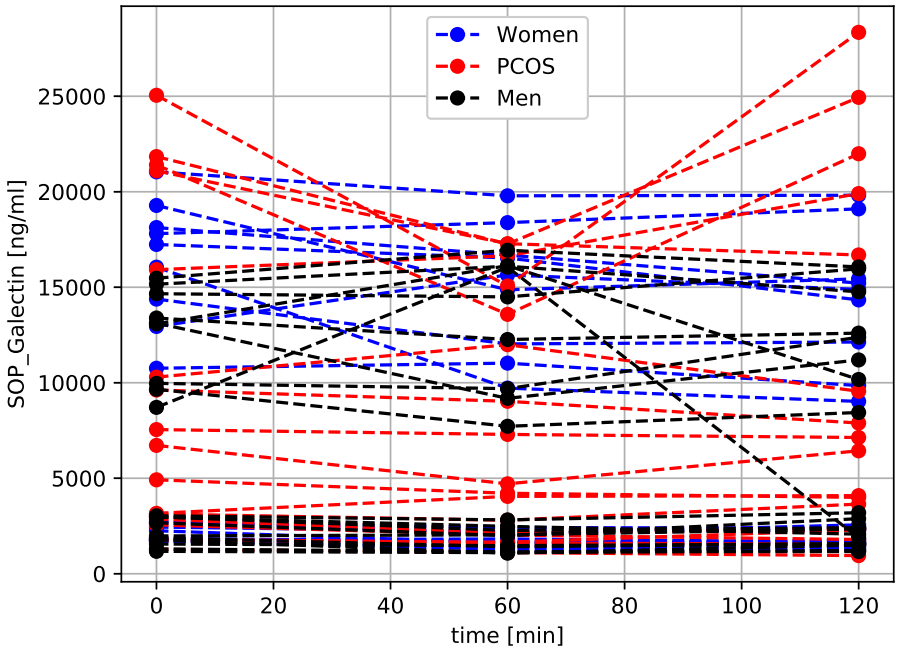

Figure 5.10: Data as a function of time for all patients, SOP_Galectin

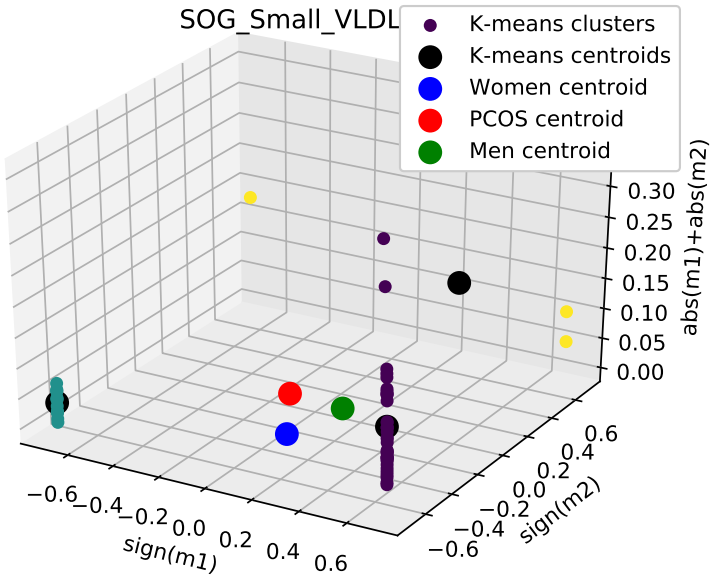

Figure 5.11: K-means algorithm results and group centroids for SOG_Small_VLDL_P

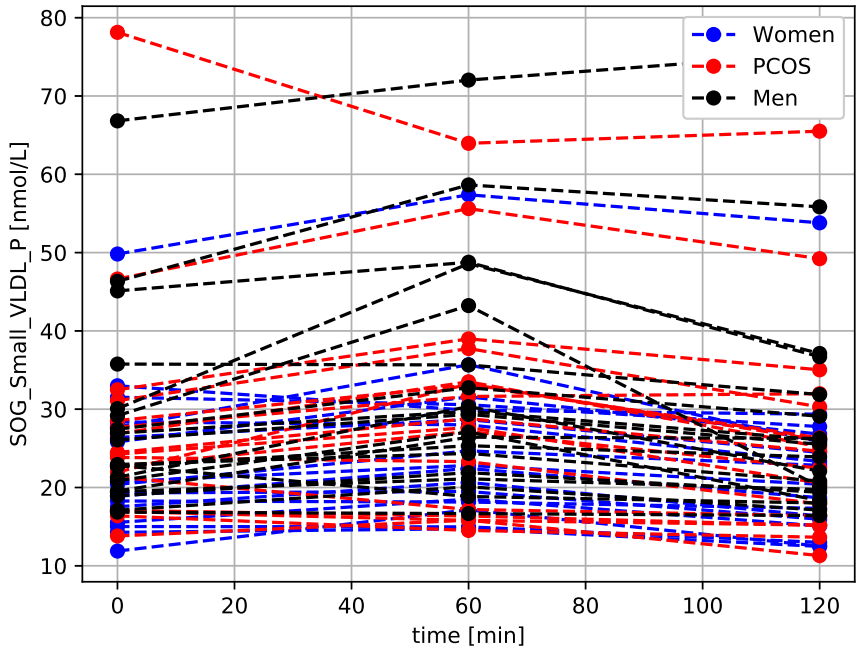

Figure 5.12: Data as a function of time for all patients, SOG_Small_VLDL_P

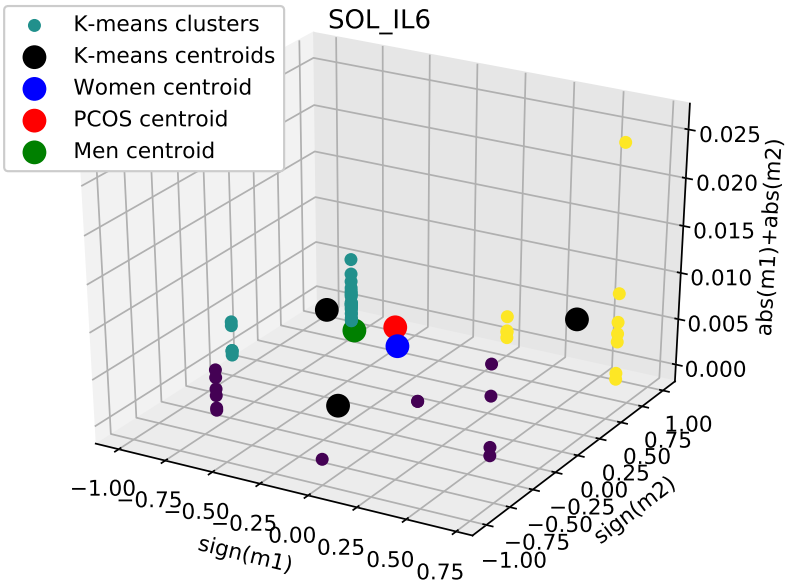

Figure 5.13: K-means algorithm results and group centroids for SOL_IL6

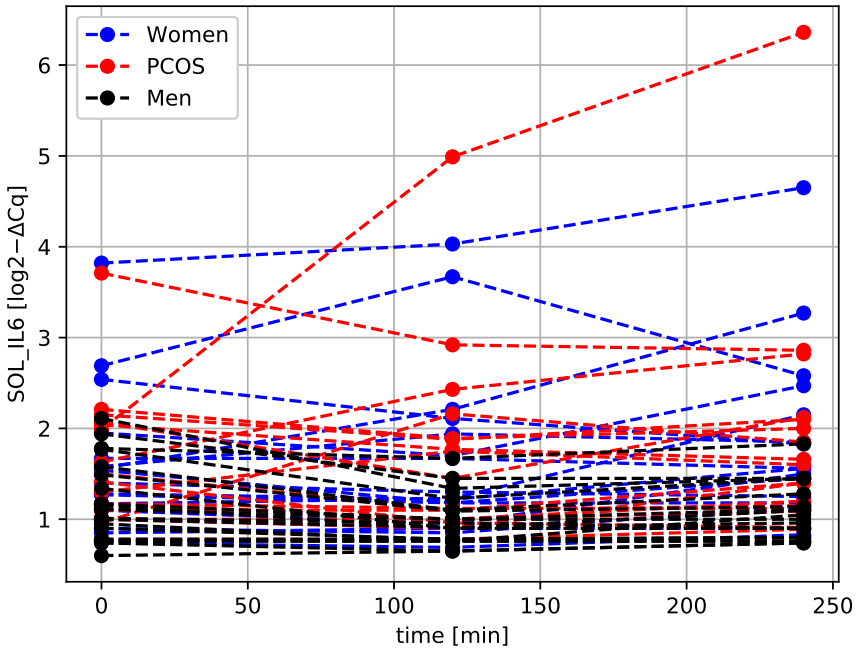

Figure 5.14: Data as a function of time for all patients, SOL_IL6

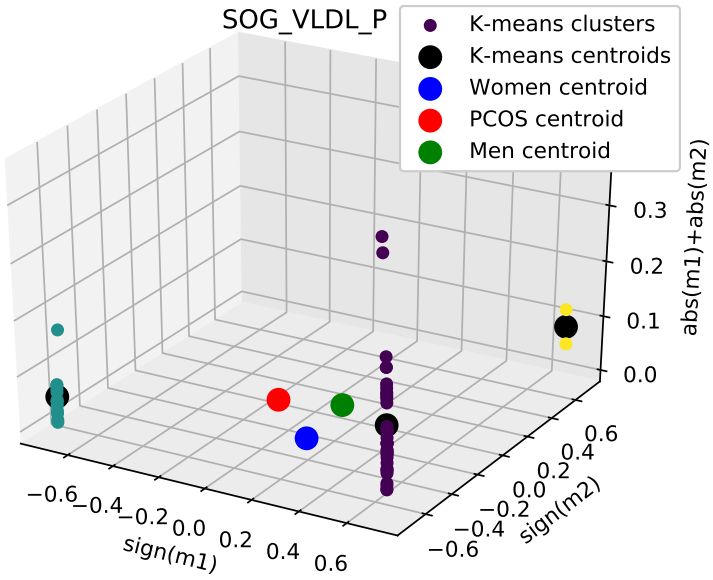

Figure 5.15: K-means algorithm results and group centroids for SOG_VLDL_P

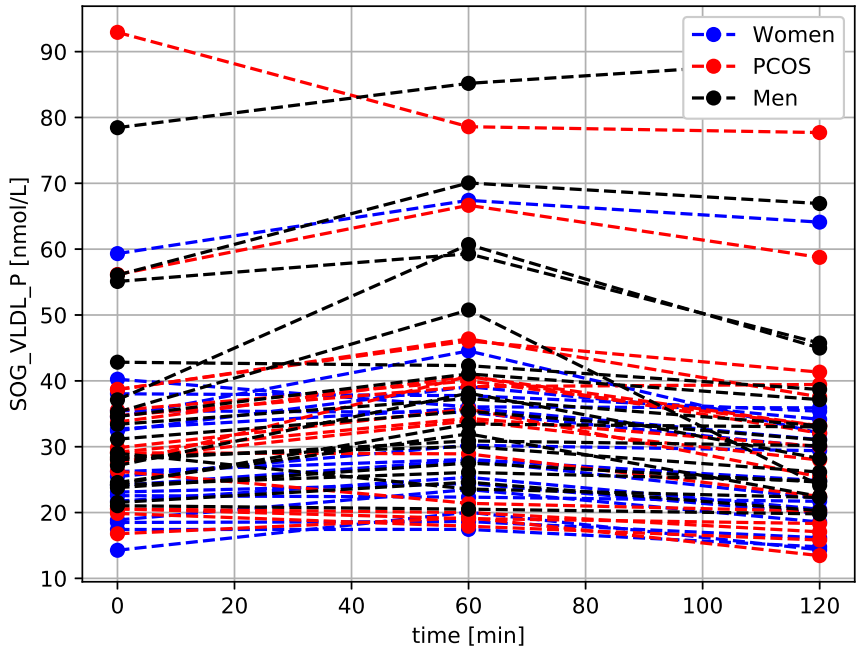

Figure 5.16: Data as a function of time for all patients, SOG_VLDL_P

Variables with defined pattern in glucose load class

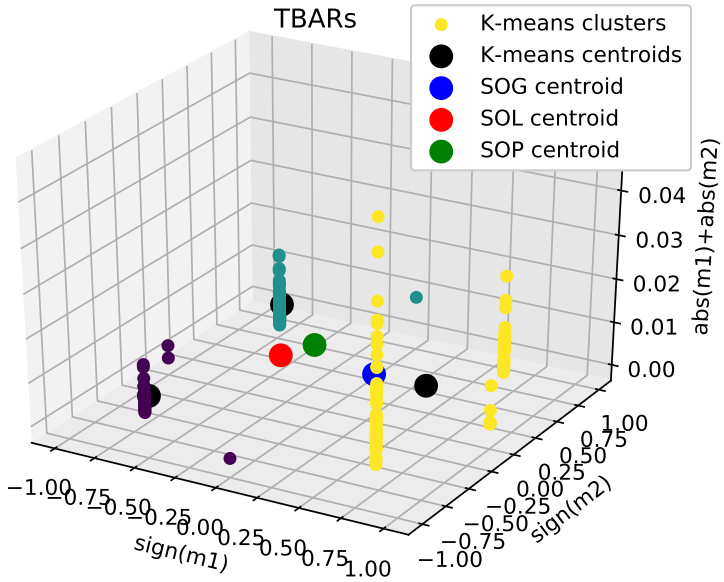

Figure 5.17: K-means algorithm results and group centroids for TBARs

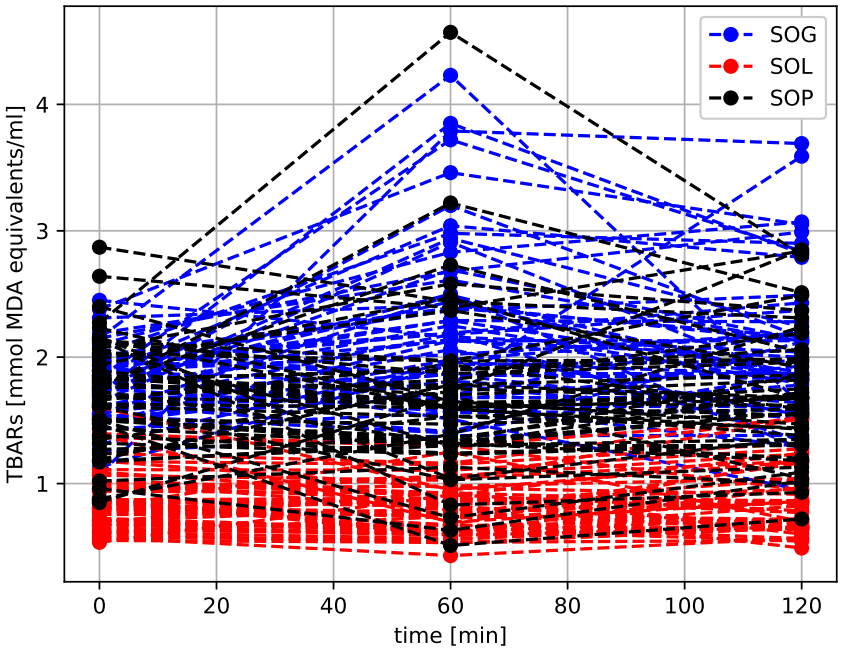

Figure 5.18: Data as a function of time for all samples, TBARs

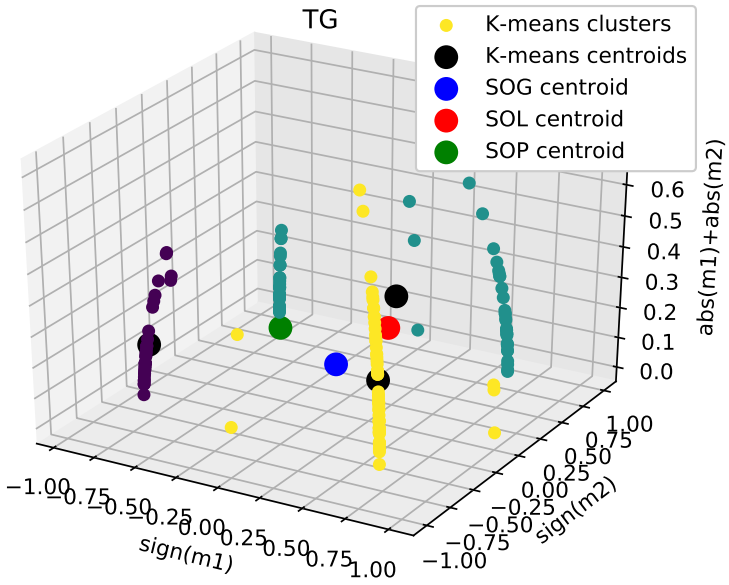

Figure 5.19: K-means algorithm results and group centroids for TG

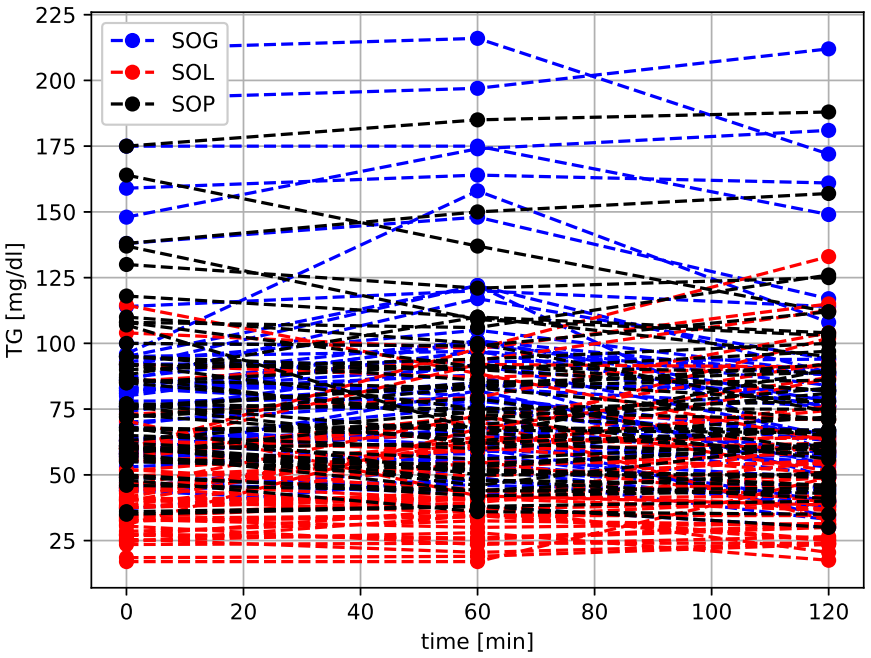

Figure 5.20: Data as a function of time for all samples, TG

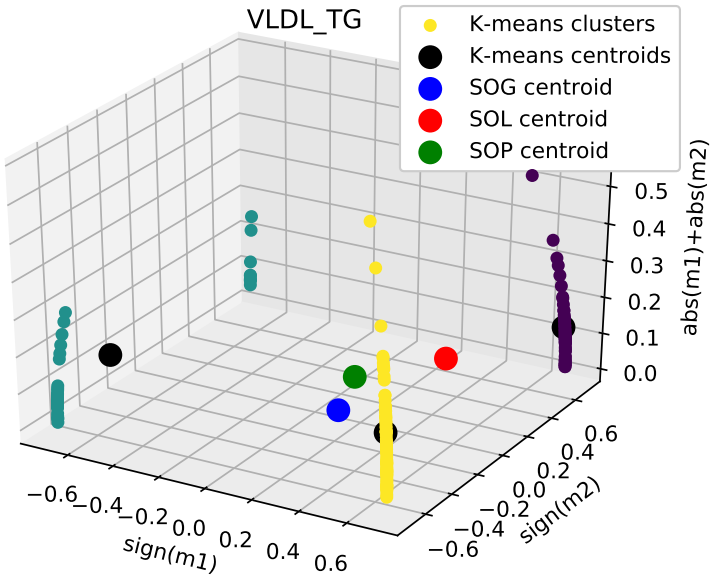

Figure 5.21: K-means algorithm results and group centroids for VLDL_TG

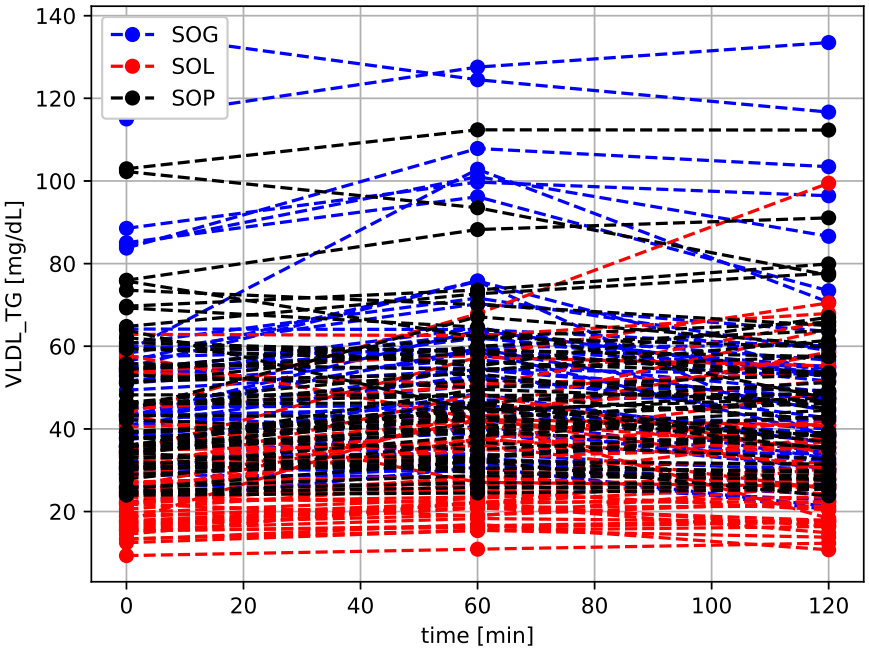

Figure 5.22: Data as a function of time for women samples, VLDL_TG

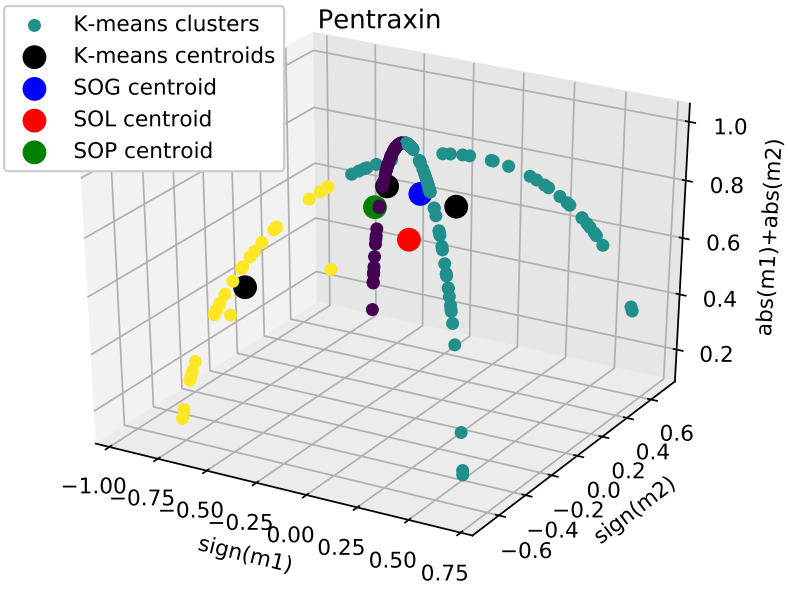

Figure 5.23: K-means algorithm results and group centroids for Pentraxin

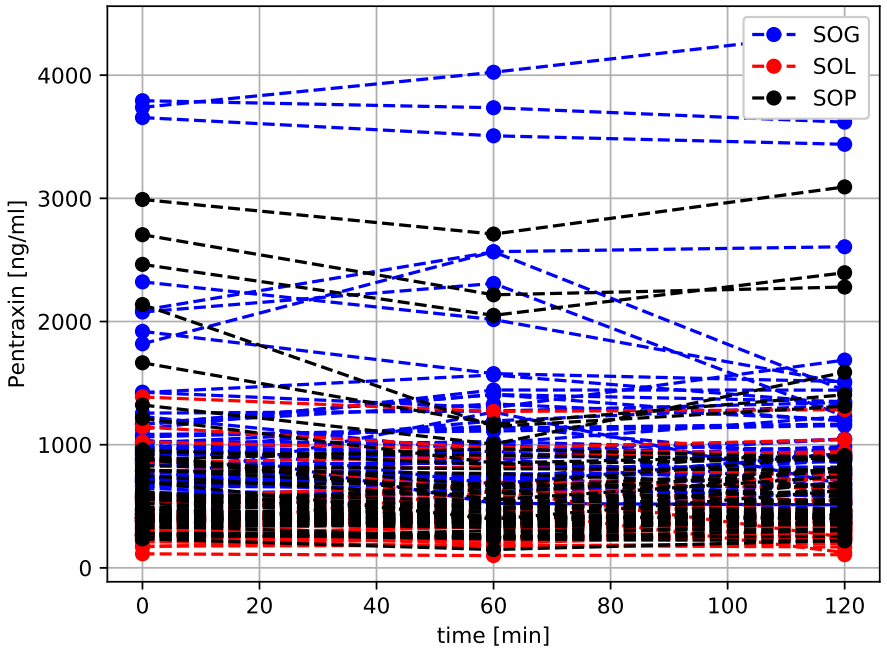

Figure 5.24: Data as a function of time for samples, Pentraxin

Variables with defined pattern in protein load class

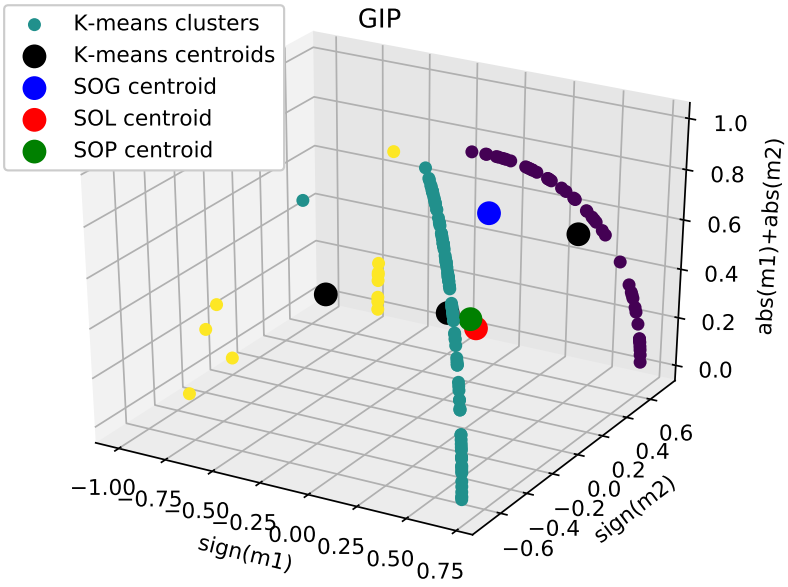

Figure 5.25: K-means algorithm results and group centroids for GIP

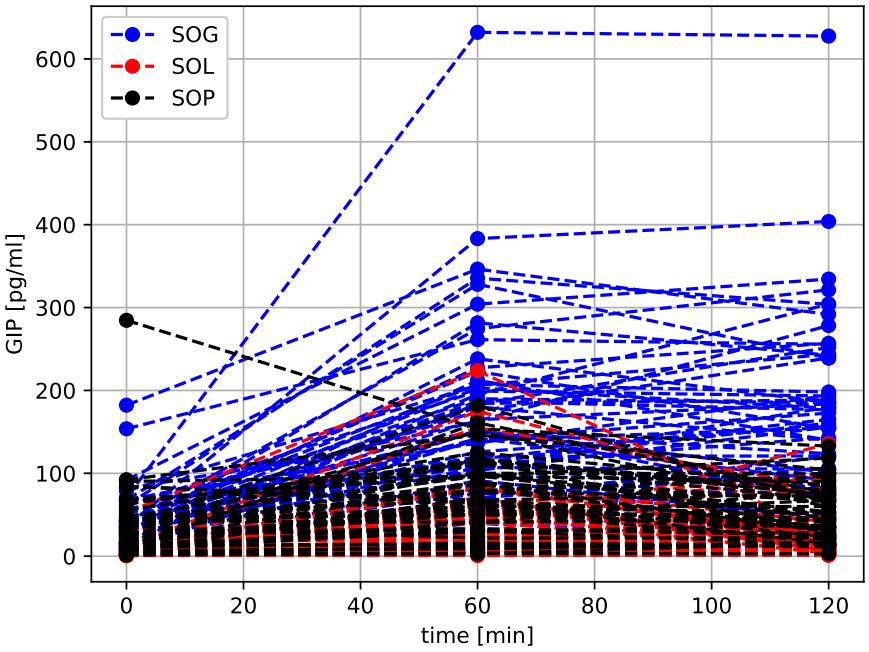

Figure 5.26: Data as a function of time for all samples, GIP

Code

The Python code used for this project are annexed in this section, as well as the output of the code in the following section.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import xlwt
from scipy.spatial.distance import seuclidean, mahalanobis
from mpl_toolkits.mplot3d import \
    Axes3D  # Though Axes3D not directly being used, it is required for 3D projection to work
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.decomposition import PCA

# ----------- Define global variables

# Choose type of group division
groups_type = 'sex'
# groups_type = 'macronutrients'

# Number of clusters in the data (independent of the number of groups that there are in the database)
num_clusters = 3  # number of classes

# Choose whether you want xls output and plot output
```

```
output_xls = 0
output_plot = 0

# Choose type of distance for centroid analysis
distance = 'euclidean'
max_dist = 0.3


def main():
    # ----------- Read xls and convert to dataframe
    xls = pd.ExcelFile('../BD tiempos TFM Maria_2.xlsx')

    # -----------  Save units of each clinical variable to dictionary
    xls3 = pd.ExcelFile('../Unidades_variables.xlsx')
    df_units = xls3.parse(xls3.sheet_names[0])
    units_collection = pd.Series(df_units.Unit.values,
                                 index=df_units.Variable1).to_dict()  # Convert certain
                                 ↪  columns of dataframe to dictionary

    # ----------- Join circulating, gene expression and liposcale databases
    # ----------- Join circulating, gene expression and liposcale databases
    df_mol = pd.read_excel(xls, 'Moléculas')
    df_ge = pd.read_excel(xls, 'Gene expression')
    df_lipo = pd.read_excel(xls, 'Liposcale')

    df_mol = df_mol.drop('Number', 1)
    df_ge = df_ge.drop('Number', 1)
    df_lipo = df_lipo.drop('Number', 1)
    df_mol = df_mol.drop('Sample', 1)
    df_ge = df_ge.drop('Sample', 1)
    df_lipo = df_lipo.drop('Sample', 1)
    df_mol = df_mol.drop('Obese', 1)
    df_ge = df_ge.drop('Obese', 1)
    df_lipo = df_lipo.drop('Obese', 1)

    df = pd.concat([df_mol, df_ge.iloc[:, 1:], df_lipo.iloc[:, 1:]], axis=1)

    # ----------- Reformulate the database if the focus is on macronutrients
    if groups_type == 'macronutrients':
        # Macronutrient reformulation
        df = df_reformulation(df)

    # ----------- Divide dataframe into groups
    group0, group1, group2, label0, label1, label2, = groups_division(df)

    # ----------- Select clinical_vars for desired number measurements
    clinical_vars = clinical_vars_selection(group0)

    # ----------- Process groups into dictionaries
    group0_collection = group_processing(clinical_vars, group0, label0 + '_',
    ↪  units_collection)
    group1_collection = group_processing(clinical_vars, group1, label1 + '_',
    ↪  units_collection)
    group2_collection = group_processing(clinical_vars, group2, label2 + '_',
    ↪  units_collection)

    # ----------- Find variables that define each group
    list1_b, list2_b, list3_b = behavior_analysis(group0_collection, group1_collection,
    ↪  group2_collection, label0,
                                                  label1, label2)
    print('Behavior')
    print(list1_b)
    print(list2_b)
```

```
84      print(list3_b)
85
86      # ----------- Principal components analysis
87      if groups_type != 'macronutrients':
88          df = features_dataframe(xls)
89          df0 = df.loc[df['GROUP'] == 0]
90          df1 = df.loc[df['GROUP'] == 1]
91          df2 = df.loc[df['GROUP'] == 2]
92          pca_analysis(df0, label0)
93          pca_analysis(df1, label1)
94          pca_analysis(df2, label2)
95
96      # ----------- Classify a new patient using 2 different classifiers
97      if groups_type != 'macronutrients':
98          xls2 = pd.ExcelFile('../BD tiempos TFM Maria_2_NUEVO_PACIENTE.xlsx')  # database
            ↪  with new patient
99          bayes_classifier(xls, xls2)  # Results with Bayes classifier
100         knn_classifier(xls, xls2)  # Results with Knn classifier
101
102     # ----------- Additional postprocessing: other plots
103     plot_all(clinical_vars, group0, group1, group2, label0, label1, label2,
        ↪  units_collection)
104
105
106 def groups_division(df):
107     # ----------- Divide input dataframe into groups
108     if groups_type == 'sex':
109         # GROUP 0 is women
110         # GROUP 1 is PCOS women
111         # GROUP 2 is men
112
113         group0 = df.loc[df['GROUP'] == 0]
114         group0 = group0.iloc[:, 1:]  # select only the columns with clinical variables
            ↪  information
115
116         group1 = df.loc[df['GROUP'] == 1]
117         group1 = group1.iloc[:, 1:]
118
119         group2 = df.loc[df['GROUP'] == 2]
120         group2 = group2.iloc[:, 1:]
121
122         # Choose labels for plots
123         label0 = 'Women'
124         label1 = 'PCOS'
125         label2 = 'Men'
126
127     elif groups_type == 'macronutrients':
128         # GROUP 0 is glucose
129         # GROUP 1 is lipids
130         # GROUP 2 is proteins
131
132         group0 = df.loc[df['GROUP'] == 0]
133         group0 = group0.iloc[:, 1:]  # select only the columns with clinical variables
            ↪  information
134
135         group1 = df.loc[df['GROUP'] == 1]
136         group1 = group1.iloc[:, 1:]
137
138         group2 = df.loc[df['GROUP'] == 2]
139         group2 = group2.iloc[:, 1:]
140
141         # Choose labels for plots
142         label0 = 'SOG'
```

```
143         label1 = 'SOL'
144         label2 = 'SOP'
145 
146     return group0, group1, group2, label0, label1, label2
147 
148 
149 def df_reformulation(df):
150     # ----------- Reformulation of macronutrients dataframe
151     df_previous = df.iloc[:, 0:1]
152 
153     df_g = df.filter(like='SOG_')
154     df_l = df.filter(
155         like='SOL_')  # SOL part of the dataframe has a bigger shape, different data
            ↪  points and more data points in some variables
156     df_p = df.filter(like='SOP_')
157 
158     # Delete some columns for SOL part (5pts reduced to 3pts in some variables to be
        ↪  able to analyze the 3 macronutrients together)
159     df_l = df_l.drop('SOL_COL_60', 1)
160     df_l = df_l.drop('SOL_COL_180', 1)
161     df_l = df_l.drop('SOL_HDL_60', 1)
162     df_l = df_l.drop('SOL_HDL_180', 1)
163     df_l = df_l.drop('SOL_LDL_60', 1)
164     df_l = df_l.drop('SOL_LDL_180', 1)
165     df_l = df_l.drop('SOL_TG_60', 1)
166     df_l = df_l.drop('SOL_TG_180', 1)
167     df_l = df_l.div(2)
168 
169     df_g.columns = df_g.columns.str.replace('SOG_', '')
170 
171     df_l.columns = df_l.columns.str.replace('SOL_', '')
172     df_l.columns = df_l.columns.str.replace('_60', '_30')
173     df_l.columns = df_l.columns.str.replace('_120', '_60')
174     df_l.columns = df_l.columns.str.replace('_180', '_90')
175     df_l.columns = df_l.columns.str.replace('_240', '_120')
176 
177     df_p.columns = df_p.columns.str.replace('SOP_', '')
178 
179     df_g = pd.concat([df_previous, df_g], axis=1)  # Concatenate two dataframes
        ↪  horizontally
180     df_l = pd.concat([df_previous, df_l], axis=1)
181     df_p = pd.concat([df_previous, df_p], axis=1)
182 
183     df_g['GROUP'] = 0
184     df_l['GROUP'] = 1
185     df_p['GROUP'] = 2
186 
187     df = pd.concat([df_g, df_l, df_p], axis=0)
188 
189     return df
190 
191 
192 def clinical_vars_selection(df):
193     # ----------- Select clinical variables from dataframe
194     header = df.columns.values.tolist()  # get dataframe header as list
195     clinical_vars = df.columns.values.tolist()  # get dataframe header as list
196 
197     for i in range(len(clinical_vars)):
198         # eliminar minutos del título
199         clinical_vars[i] = clinical_vars[i].replace('240', '')
200         clinical_vars[i] = clinical_vars[i].replace('180', '')
201         clinical_vars[i] = clinical_vars[i].replace('120', '')
202         clinical_vars[i] = clinical_vars[i].replace('90', '')
```

```
203         clinical_vars[i] = clinical_vars[i].replace('60', '')
204         clinical_vars[i] = clinical_vars[i].replace('30', '')
205         clinical_vars[i] = clinical_vars[i].replace('_0', '_')
206         clinical_vars[i] = clinical_vars[i].replace('.1',
207                                                     '')  # .1 occurrence with Omentin
                                                         ↪  (not visible in original xlsx)
208         aux_str = clinical_vars[i]
209         clinical_vars[i] = aux_str[:-1]  # remove last character from string (it's an
            ↪  underscore)
210
211     header = clinical_vars.copy()  # since if we copy a list, we don't actually have two
        ↪  lists but the same reference
212     clinical_vars = set(clinical_vars)  # unique values from list, by converting list to
        ↪  set. Not ordered alphabetically
213     clinical_vars = list(clinical_vars)  # convert the set back to a list.
214     clinical_vars = sorted(clinical_vars)  # sort list alphabetically
215
216     clinical_vars_3 = []  # initialize empty list
217     clinical_vars_5 = []
218
219     for i in range(len(clinical_vars)):
220         n = 0
221         for s in header:
222             if clinical_vars[i] == s:
223                 n += 1
224         if n == 3:
225             clinical_vars_3.append(clinical_vars[i])
226         elif n == 5:
227             clinical_vars_5.append(clinical_vars[i])
228
229     if output_xls == 1:
230         wb = xlwt.Workbook()
231         ws = wb.add_sheet('Sheet 1')
232         for i in range(len(clinical_vars_3)):
233             ws.write(i, 0, clinical_vars_3[i])
234         wb.save('./' + groups_type + '/xls/' + '3pts.xls')
235
236         wb = xlwt.Workbook()
237         ws = wb.add_sheet('Sheet 1')
238         for i in range(len(clinical_vars_5)):
239             ws.write(i, 0, clinical_vars_5[i])
240         wb.save('./' + groups_type + '/xls/' + '5pts.xls')
241
242     clinical_vars = clinical_vars_3 + clinical_vars_5
243
244     return clinical_vars
245
246
247 def group_processing(clinical_vars, df, title, units_collection):
248     # ----------- Convert dataframe into dictionary
249     df = df.replace(r'#¡NULO!', np.nan)  # Treat lost values with NaN (which is type
        ↪  float)
250
251     df = df.apply(pd.to_numeric)  # convert dataframe to numeric (because some numbers
        ↪  are imported as strings)
252
253     dataframe_collection = {}  # initialize a dictionary of dataframes. Necessary
        ↪  because depending on NaN values,
254     # not all dataframes will have the same number of rows for all clinical_vars.
255
256     for i in range(len(clinical_vars)):
257
258         df0 = df.filter(regex='^' + clinical_vars[
```

```
259             i] + '_' + '\d')  # reg ex starting with clinical_vars[i] and ending with an
                ↪  integer
260
261         df0 = df0.dropna()
262
263         time = get_time(df0, clinical_vars[i])  # time (mins)
264
265         dataframe_collection[clinical_vars[i]] = df0  # dictionary
266
267         data_array = df0.values  # df0 is converted to a numpy array
268
269         # Plots
270         if output_plot == 1:
271             m, n = data_array.shape
272             fig = plt.figure()
273             for j in range(0, m):
274                 plt.plot(time, data_array[j, :n], 'o--')
275             plt.xlabel('time [min]')
276             aux1 = clinical_vars[i]
277             if groups_type == 'sex':
278                 plt.ylabel(clinical_vars[i] + ' ' + '[' + units_collection[aux1[4:]] +
                    ↪  ']')
279             elif groups_type == 'macronutrients':
280                 plt.ylabel(clinical_vars[i] + ' ' + '[' + units_collection[aux1] + ']')
281             plt.grid(axis='both')
282             plt.title(title)
283             plt.savefig(
284                 './' + groups_type + '/plots/' + 'data/' + title + clinical_vars[i] +
                    ↪  '.eps', format='eps')
285             plt.savefig(
286                 './' + groups_type + '/plots/' + 'data/' + title + clinical_vars[i] +
                    ↪  '.png', format='png')
287             plt.close()
288
289     return dataframe_collection
290
291
292 def plot_all(clinical_vars, group0, group1, group2, label0, label1, label2,
    ↪  units_collection):
293     # ----------- Convert dataframe into dictionary
294     group0 = group0.replace(r'#¡NULO!', np.nan)  # Treat lost values with NaN (which is
        ↪  type float)
295     group0 = group0.apply(pd.to_numeric)  # convert dataframe to numeric (because some
        ↪  numbers are imported as strings)
296
297     group1 = group1.replace(r'#¡NULO!', np.nan)  # Treat lost values with NaN (which is
        ↪  type float)
298     group1 = group1.apply(pd.to_numeric)  # convert dataframe to numeric (because some
        ↪  numbers are imported as strings)
299
300     group2 = group2.replace(r'#¡NULO!', np.nan)  # Treat lost values with NaN (which is
        ↪  type float)
301     group2 = group2.apply(pd.to_numeric)  # convert dataframe to numeric (because some
        ↪  numbers are imported as strings)
302
303     for i in range(len(clinical_vars)):
304
305         df0 = group0.filter(regex='^' + clinical_vars[
306             i] + '_' + '\d')  # reg ex starting with clinical_vars[i] and ending with an
                ↪  integer
307         df0 = df0.dropna()
308         time = get_time(df0, clinical_vars[i])  # time (mins)
309         data_array_0 = df0.values  # df0 is converted to a numpy array
```

```
310
311         df1 = group1.filter(regex='^' + clinical_vars[
312             i] + '_' + '\d')  # reg ex starting with clinical_vars[i] and ending with an
                ↪  integer
313         df1 = df1.dropna()
314         data_array_1 = df1.values
315
316         df2 = group2.filter(regex='^' + clinical_vars[
317             i] + '_' + '\d')  # reg ex starting with clinical_vars[i] and ending with an
                ↪  integer
318         df2 = df2.dropna()
319         data_array_2 = df2.values
320
321         # Plot all groups in same graph
322         if output_plot == 1:
323             fig = plt.figure()
324             m0, n = data_array_0.shape
325             for j in range(0, m0):
326                 if j == 0:
327                     plt.plot(time, data_array_0[j, :n], 'bo--', label=label0)
328                 else:
329                     plt.plot(time, data_array_0[j, :n], 'bo--')
330             m1, n = data_array_1.shape
331             for j in range(0, m1):
332                 if j == 0:
333                     plt.plot(time, data_array_1[j, :n], 'ro--', label=label1)
334                 else:
335                     plt.plot(time, data_array_1[j, :n], 'ro--')
336             m2, n = data_array_2.shape
337             for j in range(0, m2):
338                 if j == 0:
339                     plt.plot(time, data_array_2[j, :n], 'ko--', label=label2)
340                 else:
341                     plt.plot(time, data_array_2[j, :n], 'ko--')
342             plt.xlabel('time [min]')
343             aux1 = clinical_vars[i]
344             if groups_type == 'sex':
345                 plt.ylabel(clinical_vars[i] + ' ' + '[' + units_collection[aux1[4:]] +
                    ↪  ']')
346             elif groups_type == 'macronutrients':
347                 plt.ylabel(clinical_vars[i] + ' ' + '[' + units_collection[aux1] + ']')
348             plt.grid(axis='both')
349             plt.legend()
350             plt.savefig(
351                 './' + groups_type + '/plots/' + 'data/' + 'All_' + clinical_vars[i] +
                    ↪  '.eps', format='eps')
352             plt.savefig(
353                 './' + groups_type + '/plots/' + 'data/' + 'All_' + clinical_vars[i] +
                    ↪  '.png', format='png')
354             plt.close()
355
356
357 def get_time(df0, key):
358     # Get time (mins) from column header of dataframe
359     df1_header = df0.columns.values.tolist()  # get dataframe header as list
360     time = np.zeros([len(df1_header)])  # initialize
361     for j in range(len(df1_header)):
362         aux_str = df1_header[j]
363         aux_str = aux_str[len(key + '_'):]
364         time[j] = int(aux_str)  # Convert string to integer
365
366     return time
367
```

```
368
369 def preliminary_analysis(group0_collection, group1_collection, group2_collection):
370     # ----------- Analysis of groups exactly as they are in database (known outputs)
371     list1 = []
372     list2 = []
373     list3 = []
374
375     for key in group0_collection.keys():
376
377         df0 = group0_collection[key]
378         df1 = group1_collection[key]
379         df2 = group2_collection[key]
380         time = get_time(df0, key)  # time (mins)
381
382         data_group0 = trend_features(df0, time)
383         data_group1 = trend_features(df1, time)
384         data_group2 = trend_features(df2, time)
385
386         kmeans = KMeans(n_clusters=1)
387         kmeans.fit(data_group0)
388         centers_orig = kmeans.cluster_centers_
389
390         kmeans = KMeans(n_clusters=1)
391         kmeans.fit(data_group1)
392         centers_orig = np.vstack([centers_orig, kmeans.cluster_centers_])
393
394         kmeans = KMeans(n_clusters=1)
395         kmeans.fit(data_group2)
396         centers_orig = np.vstack([centers_orig, kmeans.cluster_centers_])
397
398         u0 = centers_orig[0, :].T
399         u1 = centers_orig[1, :].T
400         u2 = centers_orig[2, :].T
401
402         # Preliminary analysis for specific behavior of women
403
404         d1 = np.linalg.norm(u1 - u2)
405         kmeans = KMeans(n_clusters=1)
406         kmeans.fit(np.vstack([u1, u2]))
407         center_u1u2 = kmeans.cluster_centers_
408         d2 = np.linalg.norm(center_u1u2 - u0)
409         if d2 > (5 * d1):
410             list1.append(key)
411
412         # Preliminary analysis for specific behavior of PCOS women
413
414         d1 = np.linalg.norm(u0 - u2)
415         kmeans = KMeans(n_clusters=1)
416         kmeans.fit(np.vstack([u0, u2]))
417         center_u0u2 = kmeans.cluster_centers_
418         d2 = np.linalg.norm(center_u0u2 - u1)
419         if d2 > (5 * d1):
420             list2.append(key)
421
422         # Preliminary analysis for specific behavior of men
423
424         d1 = np.linalg.norm(u0 - u1)
425         kmeans = KMeans(n_clusters=1)
426         kmeans.fit(np.vstack([u0, u1]))
427         center_u0u1 = kmeans.cluster_centers_
428         d2 = np.linalg.norm(center_u0u1 - u2)
429         if d2 > (5 * d1):
430             list3.append(key)
```

```
431
432     return list1, list2, list3
433
434
435 def behavior_analysis(group0_collection, group1_collection, group2_collection, label0,
    ↪  label1, label2):
436     # ----------- Get variables that define each group based on their original centroids
        ↪  and K-means centroids
437     list1 = []
438     list2 = []
439     list3 = []
440
441     for key in group0_collection.keys():
442
443         df0 = group0_collection[key]
444         df1 = group1_collection[key]
445         df2 = group2_collection[key]
446         time = get_time(df0, key)  # time (mins)
447
448         data_group0 = trend_features(df0, time)
449         data_group1 = trend_features(df1, time)
450         data_group2 = trend_features(df2, time)
451
452         kmeans = KMeans(n_clusters=1)
453         kmeans.fit(data_group0)
454         centers_orig = kmeans.cluster_centers_
455
456         kmeans = KMeans(n_clusters=1)
457         kmeans.fit(data_group1)
458         centers_orig = np.vstack([centers_orig, kmeans.cluster_centers_])
459
460         kmeans = KMeans(n_clusters=1)
461         kmeans.fit(data_group2)
462         centers_orig = np.vstack([centers_orig, kmeans.cluster_centers_])
463
464         u0 = centers_orig[0, :].T
465         u1 = centers_orig[1, :].T
466         u2 = centers_orig[2, :].T
467
468         # Determine k-means centroids
469
470         X = np.concatenate((data_group0, data_group1, data_group2))
471
472         centers, y_kmeans = kmeans_clustering(X)
473
474         # Behavior analysis comparing original centroids with k-means centroids
475
476         dist_group0 = np.zeros([num_clusters])
477         dist_group1 = np.zeros([num_clusters])
478         dist_group2 = np.zeros([num_clusters])
479
480         for i in np.arange(0, num_clusters):
481
482             v = centers[i, :]
483
484             if distance == 'euclidean':
485                 # Euclidean distance
486                 dist_group0[i] = np.linalg.norm(u0 - v)
487                 dist_group1[i] = np.linalg.norm(u1 - v)
488                 dist_group2[i] = np.linalg.norm(u2 - v)
489             elif distance == 'seuclidean':
490                 # Normalized Euclidean distance
491                 U1 = np.vstack((u0, v))
```

```
                var_U1 = np.var(U1.T, axis=1)  # component variances
                dist_group0[i] = seuclidean(u0, v, var_U1)
                U2 = np.vstack((u1, v))
                var_U2 = np.var(U2.T, axis=1)
                dist_group1[i] = seuclidean(u1, v, var_U2)
                U3 = np.vstack((u2, v))
                var_U3 = np.var(U3.T, axis=1)
                dist_group2[i] = seuclidean(u2, v, var_U3)
            elif distance == 'mahalanobis':
                # singular matrix problems--> do not use
                U1 = np.vstack((u0, v))
                cov_uv1 = np.cov(U1.T)
                icov_uv1 = np.linalg.inv(cov_uv1)
                dist_group0[i] = mahalanobis(u0, v, icov_uv1)
                U2 = np.vstack((u1, v))
                cov_uv2 = np.cov(U2.T)
                icov_uv2 = np.linalg.inv(cov_uv2)
                dist_group1[i] = mahalanobis(u1, v, icov_uv2)
                U3 = np.vstack((u2, v))
                cov_uv3 = np.cov(U3.T)
                icov_uv3 = np.linalg.inv(cov_uv3)
                dist_group2[i] = mahalanobis(u2, v, icov_uv3)

            if dist_group0[i] <= max_dist and dist_group1[i] > max_dist and
            ↪  dist_group2[i] > max_dist:
                list1.append(key)
            elif dist_group0[i] > max_dist and dist_group1[i] <= max_dist and
            ↪  dist_group2[i] > max_dist:
                list2.append(key)
            elif dist_group0[i] > max_dist and dist_group1[i] > max_dist and
            ↪  dist_group2[i] <= max_dist:
                list3.append(key)

        # Plot clusters and centroids

        if output_plot == 1 and np.shape(centers)[1] == 3:  # only plot with 3 points, 5
        ↪  points clusters are 6D
            fig = plt.figure()
            ax = fig.add_subplot(111, projection='3d')

            ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=y_kmeans, s=25, cmap='viridis',
            ↪  label='K-means clusters')
            ax.scatter(centers[:, 0], centers[:, 1], centers[:, 2], c='black', s=100,
            ↪  alpha=0.5,
                       label='K-means centroids')
            ax.scatter(centers_orig[0, 0], centers_orig[0, 1], centers_orig[0, 2],
            ↪  c='blue', s=100, alpha=0.5,
                       label=label0 + ' centroid')
            ax.scatter(centers_orig[1, 0], centers_orig[1, 1], centers_orig[1, 2],
            ↪  c='red', s=100, alpha=0.5,
                       label=label1 + ' centroid')
            ax.scatter(centers_orig[2, 0], centers_orig[2, 1], centers_orig[2, 2],
            ↪  c='green', s=100, alpha=0.5,
                       label=label2 + ' centroid')

            ax.set_xlabel('sign(m1)')
            ax.set_ylabel('sign(m2)')
            ax.set_zlabel('abs(m1)+abs(m2)')

            plt.title(key)
            plt.legend()
            plt.grid(axis='both')
```

```
545             plt.savefig('./' + groups_type + '/plots/' + 'clustering/' + key + '.eps',
                ↪  format='eps')
546             plt.savefig('./' + groups_type + '/plots/' + 'clustering/' + key + '.png',
                ↪  format='png')
547             plt.cla()
548             plt.clf()
549             plt.close()
550
551     return set(list1), set(list2), set(list3)
552
553
554 def kmeans_clustering(x):
555     # ----------- Perform K-means clustering for n_clusters on data series
556     kmeans = KMeans(n_clusters=num_clusters)
557     kmeans.fit(x)
558     y_kmeans = kmeans.predict(x)
559     centers = kmeans.cluster_centers_
560
561     return centers, y_kmeans
562
563
564 def trend_features(df, time):
565     # ----------- Get features for each time series
566     if len(time) == 3:
567
568         data_df = df.values  # df is converted to a numpy array
569
570         m, n = data_df.shape
571         trend_data = np.zeros(
572             [m, 3])  # 3 variables are added to the measurements of each patient
573
574         for j in range(0, m):
575
576             m1 = (data_df[j, 1] - data_df[j, 0]) / (time[1] - time[0])
577             m2 = (data_df[j, 2] - data_df[j, 1]) / (time[2] - time[1])
578
579             trend_data[j, 0] = np.sign(m1)
580             trend_data[j, 1] = np.sign(m2)
581             trend_data[j, 2] = abs(m1) + abs(m2)
582
583             # Normalize results vector. Warning: there are some cases where there is no
                ↪  reaction, if is written
584             norm = np.linalg.norm(trend_data[j, :])
585             if data_df[j, 0] == data_df[j, 1] and data_df[j, 0] == data_df[j, 2]:
586                 trend_data[j, :] = [0, 0, 0]
587             else:
588                 trend_data[j, :] = np.divide(trend_data[j, :], norm)
589
590     elif len(time) == 5:
591         data_df = np.log10(df.values)  # df is converted to a numpy array
592
593         m, n = data_df.shape
594         trend_data = np.zeros(
595             [m,
596              6])  # 6 variables are added to the measurements of each patient
597
598         for j in range(0, m):
599
600             m1 = (data_df[j, 1] - data_df[j, 0]) / (time[1] - time[0])
601             m2 = (data_df[j, 2] - data_df[j, 1]) / (time[2] - time[1])
602             m3 = (data_df[j, 3] - data_df[j, 2]) / (time[3] - time[2])
603             m4 = (data_df[j, 4] - data_df[j, 3]) / (time[4] - time[3])
604
```

```
605             trend_data[j, 0] = np.sign(m1)
606             if data_df[j, 1] == data_df[j, 0]:
607                 trend_data[j, 0] = 0
608
609             trend_data[j, 1] = np.sign(m2)
610             if data_df[j, 2] == data_df[j, 1]:
611                 trend_data[j, 1] = 0
612
613             trend_data[j, 2] = np.sign(m3)
614             if data_df[j, 3] == data_df[j, 2]:
615                 trend_data[j, 2] = 0
616
617             trend_data[j, 3] = np.sign(m4)
618             if data_df[j, 4] == data_df[j, 3]:
619                 trend_data[j, 3] = 0
620
621             p = np.polyfit(time, data_df[j, 0:n], 3)  # polyfit 3rd degree. Highest
                ↪  power first
622             trend_data[j, 4] = abs(p[1])  # quadratic coefficient
623             trend_data[j, 5] = abs(p[2])  # linear coefficient
624
625             # Normalize results vector
626             norm = np.linalg.norm(trend_data[j, :])
627             trend_data[j, :] = np.divide(trend_data[j, :], norm)
628
629     return trend_data
630
631
632 def pca_analysis(df, label0):
633     # -----------  Principal components analysis
634     df.columns = df.columns.str.replace('_norm', '')
635     X = df.drop('GROUP', 1)
636     y = df['GROUP']
637     names = list(X.columns.values)
638
639     # Scale the data so that each feature has unit variance
640     sc = StandardScaler()
641     X_scaled = sc.fit_transform(X)
642
643     # PCA transform and select number of components that explain most of the variance
        ↪  (>80% of the variance)
644     pca = PCA(n_components=11)  # cant be higher than 47 (number of patients)
645     pca.fit(X_scaled)
646     X_pca = pca.transform(X_scaled)  # features size is now n_components
647
648     list1 = []
649     list2 = []
650     for i in np.arange(0, 11):
651         components = pca.components_
652         index_max = np.argmax(abs(components[i, :]))
653         list1.append(names[index_max])
654         list2.append(components[i, index_max])
655
656     print('Result from PCA analysis')
657     print(list1)
658
659     if output_plot == 1:
660         # Plot explained variance
661         pca = PCA(n_components=13)  # cant be higher than 47 (number of patients)
662         pca.fit(X_scaled)
663         plt.figure()
664         plt.plot(np.cumsum(pca.explained_variance_ratio_), 'k--o')
665         plt.xlabel('Number of Components')
```

```
666         plt.ylabel('Explained variance ratio')  # for each component
667         plt.title(label0)
668         plt.grid()
669         plt.savefig(
670             './' + groups_type + '/plots/' + 'pcas/' + label0 + '.eps', format='eps')
671         plt.savefig(
672             './' + groups_type + '/plots/' + 'pcas/' + label0 + '.png', format='png')
673
674
675 def bayes_classifier(xls, xls2):
676     # ----------- Train and test a Gaussian Naive Bayes classifier with our data for sex
        ↪ groups
677     df = features_dataframe(xls)
678
679     # Perform classification
680
681     # Split dataset in training and test datasets
682     X_train, X_test = train_test_split(df, test_size=0.3, random_state=0)  # 70%
        ↪ training and 30% test
683
684     # Instantiate the classifier
685     gnb = GaussianNB()
686
687     # Train classifier
688     gnb1 = gnb.fit(X_train.values, X_train['GROUP'])  # Train the classifier with all
        ↪ clinical variables
689
690     # Predict the response for test dataset
691     y_test = X_test['GROUP'].values
692     y_pred = gnb1.predict(X_test)
693
694     # See how many results are right in percentage
695     accuracy = metrics.accuracy_score(y_test, y_pred) * 100
696
697     print('Test set results')
698     print(y_pred)
699     print('Classifier performance: {:05.2f}%'.format(accuracy))
700
701     X_new = features_dataframe(xls2)
702
703     print('New patient results')
704     y = gnb1.predict(X_new)
705
706     print(y)
707
708
709 def knn_classifier(xls, xls2):
710     # ----------- Train and test a K nearest neighbors classifier with our data for sex
        ↪ groups
711     df = features_dataframe(xls)
712
713     # Perform classification
714
715     # Split dataset in training and test datasets
716     X = df.iloc[:, 1:].values
717     y = df['GROUP']
718     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
719                                                         random_state=0)  # 70% training
                                                            ↪ and 30% test
720
721     # Scale the variables
722     scaler = StandardScaler()
723     scaler.fit(X_train)
```

```
    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)

    # Instantiate the classifier
    knn1 = KNeighborsClassifier(n_neighbors=5)

    # Train classifier
    knn1.fit(X_train, y_train)

    # Predict the response for test dataset
    y_pred = knn1.predict(X_test)

    # See how many results are right in percentage
    accuracy = metrics.accuracy_score(y_test, y_pred) * 100

    print('Test set results')
    print(y_pred)
    print('Classifier performance: {:05.2f}%'.format(accuracy))

    df2 = features_dataframe(xls2)

    # Scale the variables
    X_new = scaler.transform(df2.values[:, 1:])

    print('New patient results')
    y = knn1.predict(X_new)

    print(y)


def trend_features_2(df, time):
    # ----------- Get features for each time series
    if len(time) == 3:

        data_df = df.values  # df is converted to a numpy array

        m, n = data_df.shape
        trend_data = np.zeros(
            [m, 1])  # 1 variable is added to the measurements of each patient

        for j in range(0, m):
            m1 = (data_df[j, 1] - data_df[j, 0]) / (time[1] - time[0])
            m2 = (data_df[j, 2] - data_df[j, 1]) / (time[2] - time[1])

            trend_data[j, 0] = np.linalg.norm(
                np.array([np.sign(m1), np.sign(m2), abs(m1) + abs(m2)]))  # norm of the
                ↪  features vector

    elif len(time) == 5:
        data_df = np.log10(df.values)  # df is converted to a numpy array

        m, n = data_df.shape
        trend_data = np.zeros(
            [m,
             1])  # 1 variable is added to the measurements of each patient

        for j in range(0, m):
            m1 = (data_df[j, 1] - data_df[j, 0]) / (time[1] - time[0])
            m2 = (data_df[j, 2] - data_df[j, 1]) / (time[2] - time[1])
            m3 = (data_df[j, 3] - data_df[j, 2]) / (time[3] - time[2])
            m4 = (data_df[j, 4] - data_df[j, 3]) / (time[4] - time[3])

```

```
785             p = np.polyfit(time, data_df[j, 0:n], 3)  # polyfit 3rd degree. Highest
                ↪ power first
786
787             trend_data[j, 0] = np.linalg.norm(
788                 np.array([np.sign(m1), np.sign(m2), np.sign(m3), np.sign(m4),
789                           abs(p[1]) + abs(p[2])]))  # norm of the features vector
790
791     return trend_data
792
793
794 def features_dataframe(xls):
795     # ----------- Get features for each clinical variable as a dataframe
796     df_mol = pd.read_excel(xls, 'Moléculas')
797     df_ge = pd.read_excel(xls, 'Gene expression')
798     df_lipo = pd.read_excel(xls, 'Liposcale')
799
800     df_mol = df_mol.drop('Number', 1)
801     df_ge = df_ge.drop('Number', 1)
802     df_lipo = df_lipo.drop('Number', 1)
803     df_mol = df_mol.drop('Sample', 1)
804     df_ge = df_ge.drop('Sample', 1)
805     df_lipo = df_lipo.drop('Sample', 1)
806     df_mol = df_mol.drop('Obese', 1)
807     df_ge = df_ge.drop('Obese', 1)
808     df_lipo = df_lipo.drop('Obese', 1)
809
810     df = pd.concat([df_mol, df_ge.iloc[:, 1:], df_lipo.iloc[:, 1:]], axis=1)
811
812     df = df.replace(r'#¡NULO!', np.nan)  # Treat lost values with NaN (which is type
        ↪ float)
813
814     df = df.apply(pd.to_numeric)  # Convert dataframe to numeric (because some numbers
        ↪ are imported as strings)
815
816     df_group = df['GROUP']
817     df_previous = df_group
818
819     # Clinical variables list
820     clinical_vars = clinical_vars_selection(df)
821
822     for i in range(len(clinical_vars)):
823
824         df0 = df.filter(regex='^' + clinical_vars[
825             i] + '_' + '\d')  # reg ex starting with clinical_vars[i] and ending with an
                ↪ integer
826
827         df0 = df0.dropna()
828
829         time = get_time(df0, clinical_vars[i])  # time (mins)
830         features = trend_features_2(df0, time)
831
832         # Dataframe of features for each clinical variable
833         if len(time) == 3:
834
835             df_caract = pd.DataFrame(data=features,
836                                      columns=[clinical_vars[i] + '_norm'])
837             df_previous = pd.concat([df_previous, df_caract], axis=1)
838
839             df_final = df_previous
840
841         elif len(time) == 5:
842
843             df_caract = pd.DataFrame(data=features,
```

```
                                        columns=[clinical_vars[i] + '_norm'])
            df_previous = pd.concat([df_previous, df_caract], axis=1)

    # Perform a dropna because not all patients have all measurements
    df_final = df_previous
    df_final = df_final.dropna()

    return df_final


if __name__ == "__main__":
    main()
```

Code output

```
Behavior
{'SOP_Pentraxin', 'SOP_GIP', 'SOL_IDL_TG', 'SOG_GIP'}
{'SOP_IL6', 'SOP_GLP1'}
{'SOG_Small_VLDL_P', 'SOP_Galectin', 'SOG_VLDL_P', 'SOP_FGF21', 'SOL_IL6'}
Result from PCA analysis
['SOL_LDL_P_HDL_P', 'SOG_Lipocalin', 'SOL_Large_LDL_P', 'SOG_Small_HDL_P', 'SOP_GL',
↪  'SOG_COL', 'SOP_LDL_TG', 'SOG_Chemerin', 'SOG_GIP', 'SOP_FGF21', 'SOP_sLeptinR']
Result from PCA analysis
['SOG_HDL_Z', 'SOL_IL1R2', 'SOP_Medium_LDL_P', 'SOP_Large_LDL_P', 'SOG_LDL_Z',
↪  'SOP_FGF23', 'SOP_Pentraxin', 'SOL_VLDL_Z', 'SOP_TLR4', 'SOL_Small_LDL_P',
↪  'SOP_TLR2']
Result from PCA analysis
['SOG_LDL_P_HDL_P', 'SOG_TNF_ge', 'SOP_LDL_C', 'SOG_Omentin', 'SOG_FGF23', 'SOG_GHRL',
↪  'SOL_TBARs', 'SOP_IL6', 'SOL_TNFRSF1B', 'SOP_Adiponectin', 'SOL_LEPR']
Test set results
[1 1 1 0 1 0 2 1 1 0 1 1 1 2 0]
Classifier performance: 100.00%
New patient results
[2]
Test set results
[1 0 0 2 0 0 2 0 1 0 1 0 0 2 0]
Classifier performance: 53.33%
New patient results
[1]
```