CLASSIFICATION OF LAPTOP COMPUTER MARKET USING FUZZY CLUSTERING: CASE STUDY ALBANIAN MARKET

A THESIS SUBMITTED TO
THE FACULTY OF ACHITECTURE AND ENGINEERING
OF
EPOKA UNIVERSITY

BY

KEJDI DOMI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR BACHELOR DEGREE
IN COMPUTER ENGINEERING

JULY, 2021

Aproval sheet of the Thesis

This is to certify that we have read the thesis entitled **“CLASSIFICATION OF LAPTOP COMPUTER MARKET USING FUZZY CLUSTERING: CASE STUDY ALBANIAN MARKET”** and that in our opinion it is fully adequate, in scope and quality, as a thesis for Bachelor Degree of Computer Engineering.

Msc. Enea Mancellari

Date: July, 07, 2021

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Kejdi Domi

Signature:

ABSTRACT

CLASSIFICATION OF LAPTOP COMPUTER MARKET USING FUZZY CLUSTERING: CASE STUDY ALBANIAN MARKET

Domi, Kejdi
B.Sc., Department of Computer Engineering
Supervisor: M. Sc Enea Mançellari

Choosing an appropriate laptop computer is a multivariable, complex problem that has proven to be a challenge. Whether you are a bussinesman looking for a portable device capable of handling your day-to-day work, or a freshman enrolled in an architecture or computer engineering program, like myself a few years ago, the diversity of features that laptops come with in today’s market make such investment difficult. To help with solving this problem, this paper aims to give some perspective in the market for laptop computer devices in Albania. This paper uses Fuzzy clustering methods, such as Fuzzy C-Means clustering, to show how different features impact the price of the product. This paper gives the general frame of the proposed analysis model with operational steps through a case study. To reconfigure it as a generative model for producing market analysis is the goal of further research that should take into consideration the age of this paper and the everchanging nature of the market.

Keywords: Fuzzy C-means algorithm, laptop computers, market analysis, The Republic of Albania

ABSTRAKT

KLASIFIKIMI I TREGUT TË KOMPJUTERAVE LAPTOP DUKE PËRDORUR GRUPIMET FUZZY: RAST STUDIMI TREGU NË SHQIPËRI

Domi, Kejdi
B.Sc., Departmenti i Inxhinierisë Kompjuterike
Supervizor: M. Sc Enea Mançellari

Zgjedhja e një laptopi të përshtatshëm është një problem me shumë ndryshore, kompleks që e ka provuar qe mund te jetë një sfidë në vetvete. Qofshi një biznesmen që po kërkoni një paisje të lëvizshme të aftë t'ju ndihmojë në punët e juaja të ditpërditshme, ose një student arkitekture apo inxhinjerie kompjuterike, si unë para disa vitesh, diversiteti i veçorive me të cilat laptopët ne tregun e sotëm vijn, e bëjn këtë investim të vështirë. Për të ndihmuar me këtë problem, ky studim ka si qëllim te japi një pikëpamje te tregut për kompjutera laptop ne Shqipëri. Studimi përdor grupimet Fuzzy, si grupimin C-means, që të konstatojë se si veçori të ndryshme ndikojn në çmimin e produktit. Ky studim jep metodologjinë gjenerale të modelit analitik nëpërmjet hapave operacionale të një rasti studimi. Rindërtimi si një model gjenerues për prodhimin e analizave të tregut do të jetë qëllimi i studimeve të mëvonëshme që duhet të marrin parasysh kohën kur ky studim është bërë dhe natyrën gjithmonë të ndryshueshme të tregut.

Fjalët Kyce: Algoritmi Fuzzy C-means, kompjuterat laptop, analizë e tregut, Republika e Shqipërise

Dedicated to my family for their support and to a special person who never doubted on me and pushed me to do better.

ACKNOWLEDGEMENTS

I wish to express my gratitude to my advisor, MSc Enea Mancellari, for continuously offering his help throughout the whole process of writing this paper.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CPU	Central Processing Unit
GPU	Graphics Processing Unit
RAM	Random Access Memory
FL	Fuzzy Logic
FCM	Fuzzy C-Means Algorithm

CHAPTER 1

INTRODUCTION

Approximately 23,000 students enroll in universities across Albania each year. The majority of those students need to buy a computer for day-to-day use and the best choice for them is to buy a portable PC, a laptop. Every year these students go through the hassle of choosing the best laptop for them that comes in an affordable price. On top of this, people choose to buy laptops for a range of other different reasons like: personal use, business requirement, gaming, etc., and they as well have to deal with a painstakingly long process of filtering out the laptops that do not fill their requirements as well as comparing laptops according to their price to notice any patterns that imply overpricing or under pricing. This paper aims to make the decision of purchasing a laptop easier by providing the reader with enough information about the laptop market in Albania so that they can make rational decisions.

Most people know their budget and are just looking for what they can settle for. A laptop's price is dependent on a variety of features: type of CPU, type of GPU, type and size of RAM just to name a few. By knowing how these features affect the price, one can make a fair decision based on what features they need, want, or desire. This would prove very handy for those that have a tight budget, but it will also be informative for customers with a considerable budget. As many other problems where FL excells at, this is an optimization problem; that is why I thought that Fuzzy algorithms would be of best fit for this paper.

1.1. Problem Introduction

Optimization problems are as old as time. I remember studying mathematical optimization in high school: we dealt with it by using derivatives. Market analysis is a difficult optimization problem. It requires thorough consideration of all the factors that affect the price of a product. When it comes to laptop computers, these factors can be divided in many ways: a division according to hardware parts of the system and software parts, dividing according to logical components of the system (CPU, GPU, Memory, Storage, Ports, etc), or even a division according to parts on the system that can be changed or extended (virtually all software can be changed, GPU can be replaced sometimes, motherboard cannot be changed). The best division for the task at hand, that is market analysis, is dividing according to logical units/components. After you have divided the whole system in these manageable parts, we have to observe the role that these parts play in the price of the product.

In Albania, most computer related stores have a website of their own, and if they don’t have a website, they use social media to advertise their products. This made it very easy to have access to information about certain system specifications for a laptop. Also, if the website had missing information, such as clock speed of a certain CPU in gigahertz, this information was easily found on the internet. The challenging part, however, would be to organize and rework this abundance of data in a meaningful way.

1.2. Aims and Objectives

The aim of this paper is to analyze which features of a laptop affect its price, how they affect it, and how we can make the best choice when it comes to buyng laptops in Albania. The data gathered is analyzed by using FC and the output is interpreted as graphs. Color is used to differentiate between the data and legends on the graph are used accordingly.

1.3. Main Idea Behind the Paper

The market of laptop sales can be analyzed in many ways, and some of them are bound to give much more insight than this paper ever will. However, the main benefit of this research is that anyone can obtain knowledge from it without having to deal with heavy economics jargon and without having to take classes on economy. This paper is designed so it can give information in a digestible way for the public. I have to emphasise that when I started writing this thesis, the target group I had in mind was freshman students, so in a way this is my attempt at being somewhat of a helping hand for them.

CHAPTER 2

LITERATURE REVIEW

2.1. Fuzzy Logic: a brief introduction

Fuzzy logic is an alternative to binary logic. In binary logic, values are limited to two states, 0 or 1, where 0 often means not true and 1 means true. In fuzzy logic a range of truth values from 0 to 1 are allowed. In this sense, fuzzy logic is the logic underlying approximate, rather than exact, modes of reasoning. [1] It leaves room for indecisiveness in an attempt to soften a rather radical and sharp “edge” of binary logic. In binary logic there is no room for uncertainty and this makes it difficult for us to represent real world problems. On the other hand, fuzzy concepts allow the quantification of phrases like “is likely”, “is unlikely”, “possibly”, “very much”, “little”, etc., as well as the traditional phrases “will occur”, “will not occur” (Figure 1). For example, a fuzzy system might decide that “is likely” means that the event has 90% probability to happen. Combining this with a continuous function (like time), we can get descriptive statements like “It rains 90% of the time in England.” that give us a considerable load of information. In simpler words, fuzzy logic is better fit to approach real life problems as most linguistic statements that describe the probability of an event can be represented mathematically by a fuzzy system. [2]

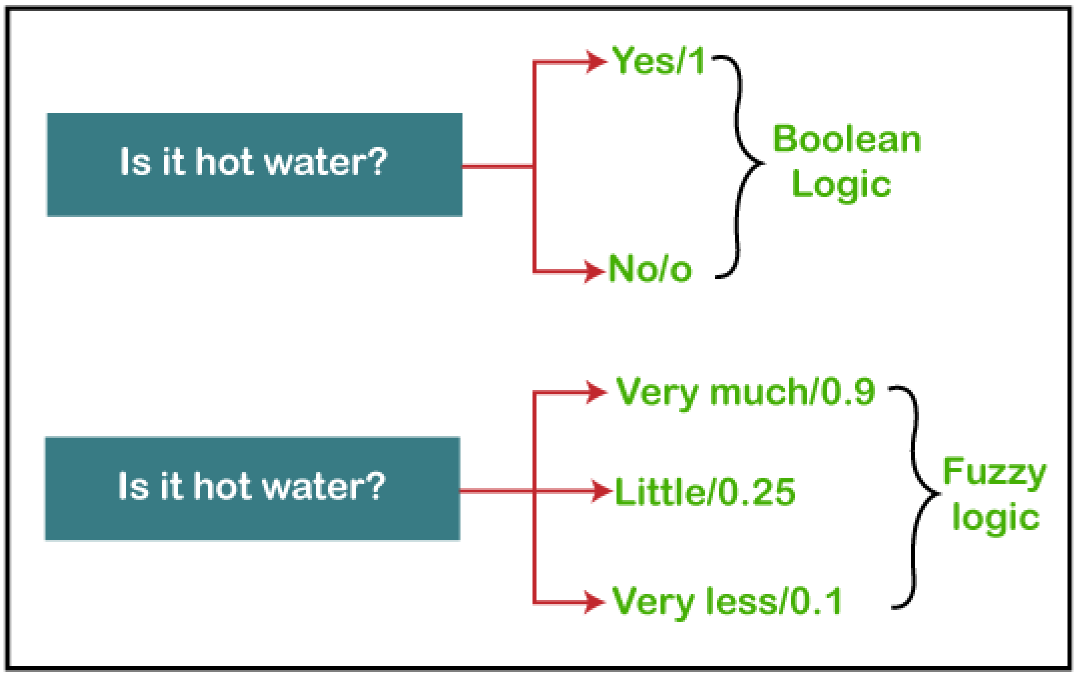


Figure 1. Fuzzy logic vs. Boolean logic

2.2. Fuzzy Clustering

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. [3] In non-fuzzy or hard clustering, data is divided into crisp clusters, where each data point belongs to exactly one cluster. In fuzzy clustering, the data points can belong to more than one cluster, and associated with each of the points are membership grades which indicate the degree to which the data points belong to the different clusters. [4] As in every clustering task the first thing to be done is to choose the number of clusters and assign coefficients to each data point at random for them being in the cluster. Then, iteratively, compute centroids and change coefficients until the change between iterations is no more than a given allowed error, **ε**.

The figure below (Figure 2) shows how fuzzy clustering (soft) is different from traditional clustering (hard). As you can see on the left, the boundary is well-defined and the data is separated in green circles and blue hexagons meant to represent the 2 classes. On the right, however, data points on each end are shown to have a high probability of being in their respective class, while data points near the meeting boundary of the 2 clusters have a probability near 0.5 for being in each class. One might say that the point in the middle of the 2 classes is 47% a green circle and 53% a blue hexagon. Figure 3 shows the pseudocode for Fuzzy C-Means Algorithm.

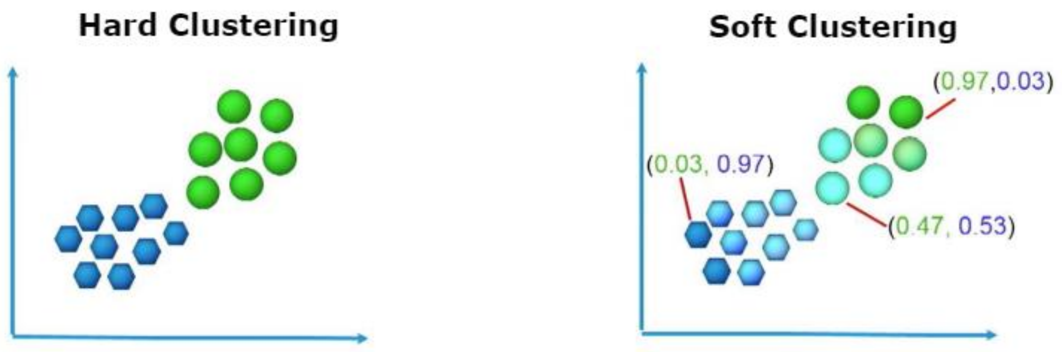
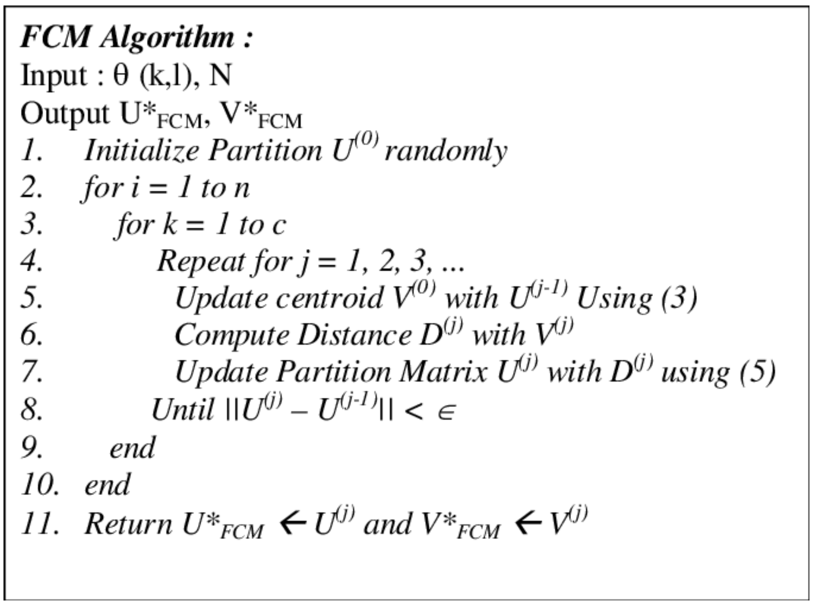


Figure 2. Traditional clustering vs. fuzzy clustering in 2-dimensional data



```
FCM Algorithm :
Input : $\theta$ (k,l), N
Output $U^*_{FCM}$, $V^*_{FCM}$
1.  Initialize Partition $U^{(0)}$ randomly
2.  for $i = 1$ to $n$
3.      for $k = 1$ to $c$
4.          Repeat for $j = 1, 2, 3, \dots$
5.              Update centroid $V^{(0)}$ with $U^{(j-1)}$ Using (3)
6.              Compute Distance $D^{(j)}$ with $V^{(j)}$
7.              Update Partition Matrix $U^{(j)}$ with $D^{(j)}$ using (5)
8.          Until $||U^{(j)} - U^{(j-1)}|| < \in$
9.      end
10. end
11. Return $U^*_{FCM} \leftarrow U^{(j)}$ and $V^*_{FCM} \leftarrow V^{(j)}$
```

Figure 3. Pseudocode of FCM

2.3. Fuzzy Concepts in Market Analysis

In 2013, a paper analyzing bank business performance and market risk was published by Yu-Chuan Chen, *et al.* The paper utilized the expanding model of Fuzzy Slack-Based Measurement to estimate market risk. The efficiency scores estimated by Fuzzy SBM model are subordinate to functional form, which provides efficiency value region in different degrees of confidence, conforms to the characteristic of risk anticipation, and estimates the management achievement of Taiwan banking under market risk. [5] Another paper used fuzzy logic to create a multi-agent e-commerce system capable of achieving a mutually beneficial deal for the seller and buyer using a negotiation process. [6]

Another paper worth noting, even though it does not use fuzzy concepts, is a research conducted by Siahaan in 2016 which had as aim the creation of a decision support system in selecting the appropriate laptop using Simple Additive Weighting. The way the data is categorized and studied in my research is heavily influenced by this paper. [7]

CHAPTER 3

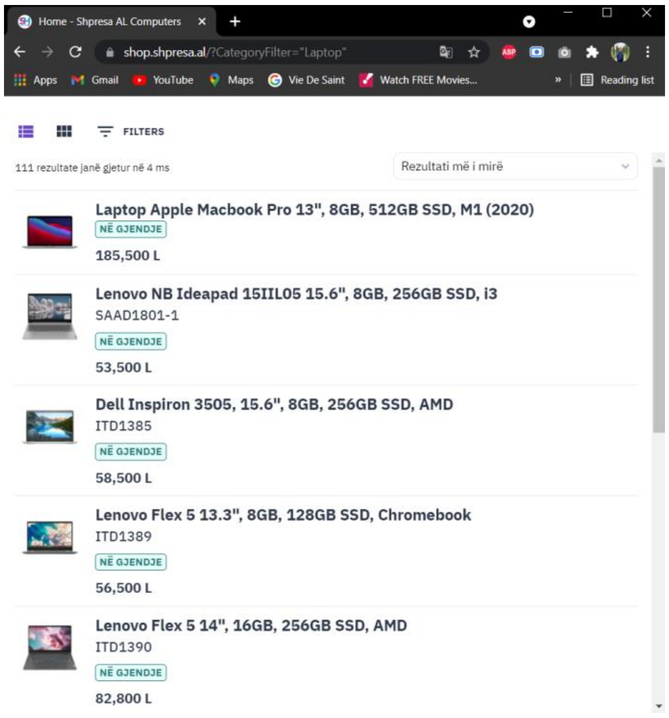
MATERIALS AND METHODOLOGY

3.1. Dataset

The dataset for this paper is composed of 70 entries of laptops and their respective specifications. It was collected from shpresa.al website (https://shop.shpresa.al/), a store located in Tirana, Albania that offers online ordering and buying of products. The reasons I chose this shop are: they are consistent with providing the specifications with each electronic device they sell, they are among the biggest shops in Albania know for selling electronic devices, the pricing is fair, and this was the shop where I bought this laptop that I am using to conduct this research. I used Excel as my database, as the collection of all data was done by me and Excel is a great tool to present data in a readable and understandable format; it also comes with great flexibility through predefined formulas or equations you can apply to your data. To process the data, retrieve the data, view the graphs that define the data and the interaction between features I used the Python programming language together with certain Python modules. To present results, I used Python and matplotlib, seaborn, and pandas modules, but more on these below.

3.1.1. Retrieving Data

My data was scattered around the web. Shpresa.al website gave a list of laptop devices when you filtered according to laptops (Figure 4). To extract the information out of 111 entries currently on the website I built a simple web crawler: it got as input the link of the page shown in figure 4 and it produced as an output an entry in the data.json file as shown below in Figure 5.



111 rezultate janë gjetur në 4 ms

Laptop Apple Macbook Pro 13", 8GB, 512GB SSD, M1 (2020)
NË GJENDJE
185,500 L

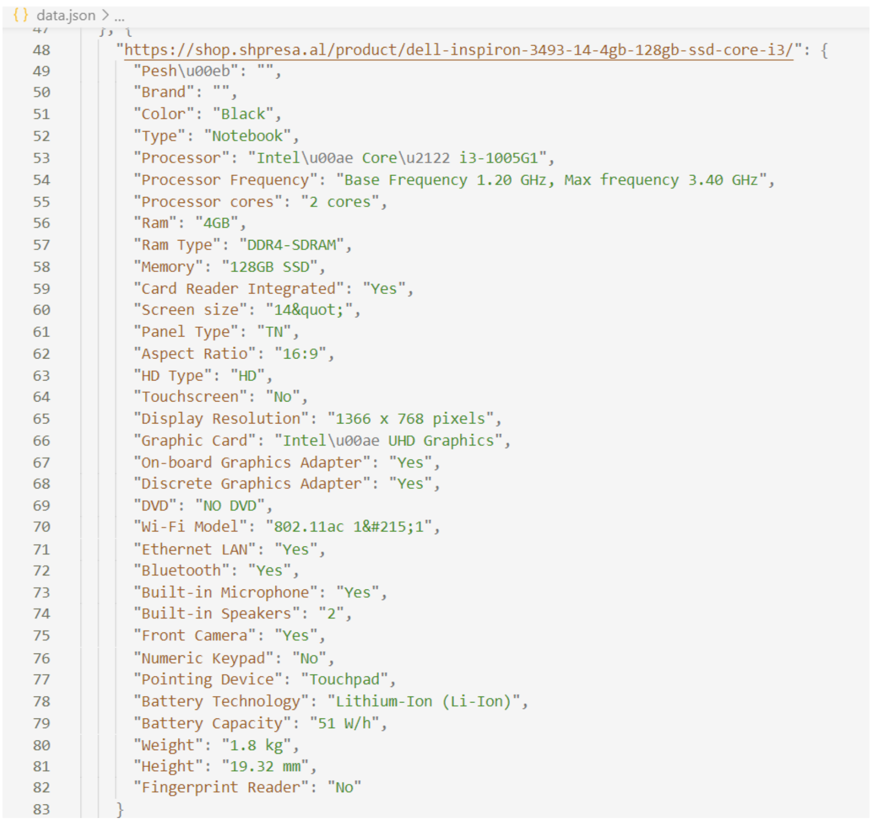
Lenovo NB Ideapad 15IIL05 15.6", 8GB, 256GB SSD, i3
SAAD1801-1
NË GJENDJE
53,500 L

Dell Inspiron 3505, 15.6", 8GB, 256GB SSD, AMD
ITD1385
NË GJENDJE
58,500 L

Lenovo Flex 5 13.3", 8GB, 128GB SSD, Chromebook
ITD1389
NË GJENDJE
56,500 L

Lenovo Flex 5 14", 16GB, 256GB SSD, AMD
ITD1390
NË GJENDJE
82,800 L

Figure 4. List of laptop devices as shown in shpresa.al website (input)



```
    "https://shop.shpresa.al/product/dell-inspiron-3493-14-4gb-128gb-ssd-core-i3/": {
        "Pesh\u00eb": "",
        "Brand": "",
        "Color": "Black",
        "Type": "Notebook",
        "Processor": "Intel\u00ae Core\u2122 i3-1005G1",
        "Processor Frequency": "Base Frequency 1.20 GHz, Max frequency 3.40 GHz",
        "Processor cores": "2 cores",
        "Ram": "4GB",
        "Ram Type": "DDR4-SDRAM",
        "Memory": "128GB SSD",
        "Card Reader Integrated": "Yes",
        "Screen size": "14&quot;",
        "Panel Type": "TN",
        "Aspect Ratio": "16:9",
        "HD Type": "HD",
        "Touchscreen": "No",
        "Display Resolution": "1366 x 768 pixels",
        "Graphic Card": "Intel\u00ae UHD Graphics",
        "On-board Graphics Adapter": "Yes",
        "Discrete Graphics Adapter": "Yes",
        "DVD": "NO DVD",
        "Wi-Fi Model": "802.11ac 1&#215;1",
        "Ethernet LAN": "Yes",
        "Bluetooth": "Yes",
        "Built-in Microphone": "Yes",
        "Built-in Speakers": "2",
        "Front Camera": "Yes",
        "Numeric Keypad": "No",
        "Pointing Device": "Touchpad",
        "Battery Technology": "Lithium-Ion (Li-Ion)",
        "Battery Capacity": "51 W/h",
        "Weight": "1.8 kg",
        "Height": "19.32 mm",
        "Fingerprint Reader": "No"
    }
```

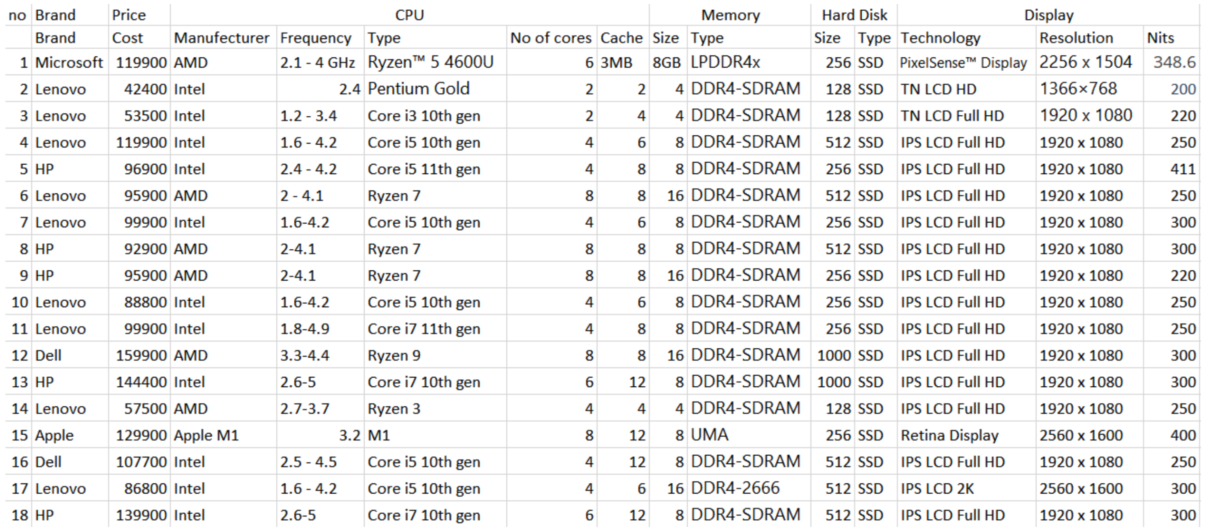
Figure 5. JSON object representing a laptop (output)

Some data shown in Figure 5 is not relevant. For example, most computers nowadays have no DVD reader. Some other data is missing, like the brand for this particular model. Out of 111 entries, I went through 70 of them, transferring the data into an Excel file, filling in the missing data and ignoring some irrelevant fields. I decided to divide the features into 13 categories, each having 1 to 5 subcategories. The division is shown in Table 1.

Table 1. Initial separation of features

No.	No.
Brand	Brand
Price	Cost
CPU	Manufecterer
	Frequency
	Type
	No of cores
	Cache
Memory	Size
	Type
Hard Disk	Size
	Type
Display	Technology
	Resolution
	Nits
Screen Size	Width
Graphics	Graphic Card
	Memory
	Memory type
	On board graphics adapter
	Discrete graphics adapter
Power	Battery capacity
	Battery durability
	AC ADAPTER POWER
Element of surprize	Touchpad
	Camera
	OS
	Audio
Security	Fingerprint
Link	Link

The data in my Excel file initially looked like this (Figure 6):



no	Brand	Price	CPU					Memory		Hard Disk		Display		
	Brand	Cost	Manufecturer	Frequency	Type	No of cores	Cache	Size	Type	Size	Type	Technology	Resolution	Nits
1	Microsoft	119900	AMD	2.1 - 4 GHz	Ryzen™ 5 4600U	6	3MB	8GB	LPDDR4x	256	SSD	PixelSense™ Display	2256 x 1504	348.6
2	Lenovo	42400	Intel	2.4	Pentium Gold	2	2	4	DDR4-SDRAM	128	SSD	TN LCD HD	1366×768	200
3	Lenovo	53500	Intel	1.2 - 3.4	Core i3 10th gen	2	4	4	DDR4-SDRAM	128	SSD	TN LCD Full HD	1920 x 1080	220
4	Lenovo	119900	Intel	1.6 - 4.2	Core i5 10th gen	4	6	8	DDR4-SDRAM	512	SSD	IPS LCD Full HD	1920 x 1080	250
5	HP	96900	Intel	2.4 - 4.2	Core i5 11th gen	4	8	8	DDR4-SDRAM	256	SSD	IPS LCD Full HD	1920 x 1080	411
6	Lenovo	95900	AMD	2 - 4.1	Ryzen 7	8	8	16	DDR4-SDRAM	512	SSD	IPS LCD Full HD	1920 x 1080	250
7	Lenovo	99900	Intel	1.6-4.2	Core i5 10th gen	4	6	8	DDR4-SDRAM	256	SSD	IPS LCD Full HD	1920 x 1080	300
8	HP	92900	AMD	2-4.1	Ryzen 7	8	8	8	DDR4-SDRAM	512	SSD	IPS LCD Full HD	1920 x 1080	300
9	HP	95900	AMD	2-4.1	Ryzen 7	8	8	16	DDR4-SDRAM	256	SSD	IPS LCD Full HD	1920 x 1080	220
10	Lenovo	88800	Intel	1.6-4.2	Core i5 10th gen	4	6	8	DDR4-SDRAM	256	SSD	IPS LCD Full HD	1920 x 1080	250
11	Lenovo	99900	Intel	1.8-4.9	Core i7 11th gen	4	8	8	DDR4-SDRAM	256	SSD	IPS LCD Full HD	1920 x 1080	250
12	Dell	159900	AMD	3.3-4.4	Ryzen 9	8	8	16	DDR4-SDRAM	1000	SSD	IPS LCD Full HD	1920 x 1080	300
13	HP	144400	Intel	2.6-5	Core i7 10th gen	6	12	8	DDR4-SDRAM	1000	SSD	IPS LCD Full HD	1920 x 1080	300
14	Lenovo	57500	AMD	2.7-3.7	Ryzen 3	4	4	4	DDR4-SDRAM	128	SSD	IPS LCD Full HD	1920 x 1080	250
15	Apple	129900	Apple M1	3.2	M1	8	12	8	UMA	256	SSD	Retina Display	2560 x 1600	400
16	Dell	107700	Intel	2.5 - 4.5	Core i5 10th gen	4	12	8	DDR4-SDRAM	512	SSD	IPS LCD Full HD	1920 x 1080	250
17	Lenovo	86800	Intel	1.6 - 4.2	Core i5 10th gen	4	6	16	DDR4-2666	512	SSD	IPS LCD 2K	2560 x 1600	300
18	HP	139900	Intel	2.6-5	Core i7 10th gen	6	12	8	DDR4-SDRAM	512	SSD	IPS LCD Full HD	1920 x 1080	300

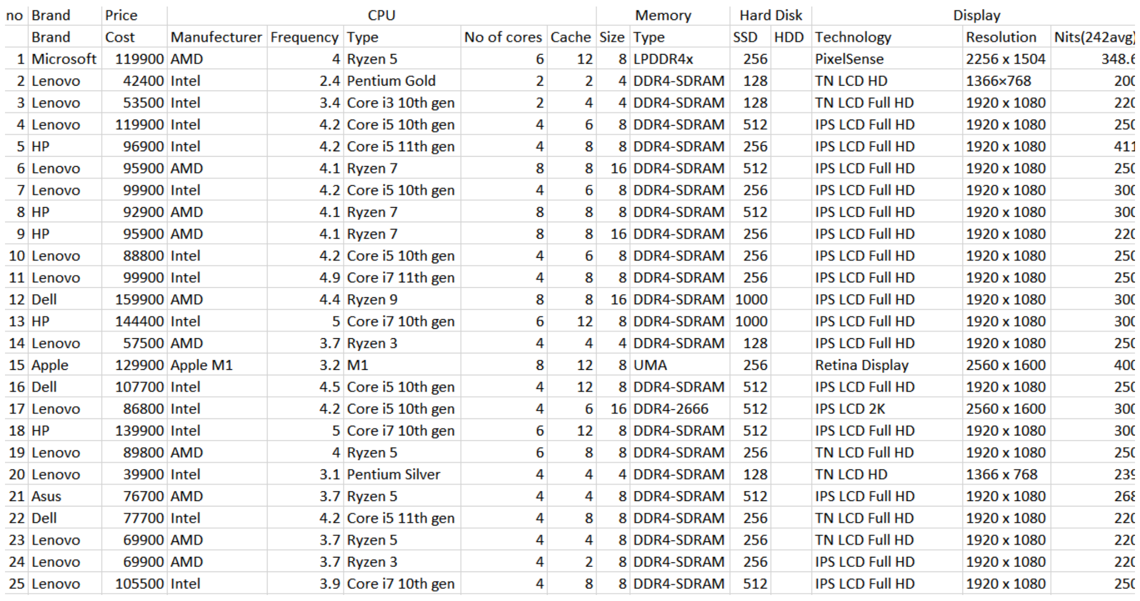
Figure 6. Representation of data in Excel

3.1.2. Data Preprocessing

This step was crucial as it removed redundancies and it prepared the data for the fuzzy algorithm implementation in Python. All of the laptops that I gathered had an SSD for storage so the Memory => Type field gives no useful information; it can be removed. For the CPU => Frequency field I saved the normal clock frequency in GHz and the max clock frequency, however, seldom only the max frequency is of importance as it is an important measure for gaming laptops (some of which are in my dataset), so I decided to use only the max frequency. After these small changes, the dataset was reduced to the one shown in Figure 7Figure 7. The final dataset had 20 columns: 1 column named “no” for indexing and 19 for the laptop features. The features are as follows:

1. brand – the brand of laptop on the dataset.
2. price – the price of laptop in LEK (will be converted later in EUR)
3. cpu_man – CPU manufacturer, many laptops had Intel as their CPU manufacturer but AMD and ARM were featured as well
4. cpu_freq – CPU frequency measured in GHz. I used top frequency.
5. cpu_type – the CPU name (i3, i5, Ryzen, etc.) and its generation.

6. n_coreas – number of cores the laptop has
7. cache_size – the size of L2 cache inside the processor
8. ram_size – the size of RAM
9. ssd_size – size of Solid-State Disk storage (secondary storage)
10. display_tech – the technology of the display that the laptop has (LED, ISP, FN, etc.)
11. px_res – the resolution in pixels
12. nits – amout of brightnesss: 1 nits = 1 candela (one candlepower) per square meter.
13. width – the width of the screen in inches
14. g_card – the GPU
15. bat_cap – battery capacity measured in Watt per hour
16. bat_dur – battery durability measured in hours
17. ac_a_power – AC adapter power measured in Watt
18. os – the operating system (Windows prevails)
19. fingerprint – whether the laptop has a fingerprint reader or not.



no	Brand	Price	CPU					Memory		Hard Disk		Display		
	Brand	Cost	Manufacturer	Frequency	Type	No of cores	Cache	Size	Type	SSD	HDD	Technology	Resolution	Nits(242avg)
1	Microsoft	119900	AMD	4	Ryzen 5	6	12	8	LPDDR4x	256		PixelSense	2256 x 1504	348.
2	Lenovo	42400	Intel	2.4	Pentium Gold	2	2	4	DDR4-SDRAM	128		TN LCD HD	1366×768	20
3	Lenovo	53500	Intel	3.4	Core i3 10th gen	2	4	4	DDR4-SDRAM	128		TN LCD Full HD	1920 x 1080	22
4	Lenovo	119900	Intel	4.2	Core i5 10th gen	4	6	8	DDR4-SDRAM	512		IPS LCD Full HD	1920 x 1080	25
5	HP	96900	Intel	4.2	Core i5 11th gen	4	8	8	DDR4-SDRAM	256		IPS LCD Full HD	1920 x 1080	411
6	Lenovo	95900	AMD	4.1	Ryzen 7	8	8	16	DDR4-SDRAM	512		IPS LCD Full HD	1920 x 1080	25
7	Lenovo	99900	Intel	4.2	Core i5 10th gen	4	6	8	DDR4-SDRAM	256		IPS LCD Full HD	1920 x 1080	30
8	HP	92900	AMD	4.1	Ryzen 7	8	8	8	DDR4-SDRAM	512		IPS LCD Full HD	1920 x 1080	30
9	HP	95900	AMD	4.1	Ryzen 7	8	8	16	DDR4-SDRAM	256		IPS LCD Full HD	1920 x 1080	22
10	Lenovo	88800	Intel	4.2	Core i5 10th gen	4	6	8	DDR4-SDRAM	256		IPS LCD Full HD	1920 x 1080	25
11	Lenovo	99900	Intel	4.9	Core i7 11th gen	4	8	8	DDR4-SDRAM	256		IPS LCD Full HD	1920 x 1080	25
12	Dell	159900	AMD	4.4	Ryzen 9	8	8	16	DDR4-SDRAM	1000		IPS LCD Full HD	1920 x 1080	30
13	HP	144400	Intel	5	Core i7 10th gen	6	12	8	DDR4-SDRAM	1000		IPS LCD Full HD	1920 x 1080	30
14	Lenovo	57500	AMD	3.7	Ryzen 3	4	4	4	DDR4-SDRAM	128		IPS LCD Full HD	1920 x 1080	25
15	Apple	129900	Apple M1	3.2	M1	8	12	8	UMA	256		Retina Display	2560 x 1600	40
16	Dell	107700	Intel	4.5	Core i5 10th gen	4	12	8	DDR4-SDRAM	512		IPS LCD Full HD	1920 x 1080	25
17	Lenovo	86800	Intel	4.2	Core i5 10th gen	4	6	16	DDR4-2666	512		IPS LCD 2K	2560 x 1600	30
18	HP	139900	Intel	5	Core i7 10th gen	6	12	8	DDR4-SDRAM	512		IPS LCD Full HD	1920 x 1080	30
19	Lenovo	89800	AMD	4	Ryzen 5	6	8	8	DDR4-SDRAM	256		TN LCD Full HD	1920 x 1080	25
20	Lenovo	39900	Intel	3.1	Pentium Silver	4	4	4	DDR4-SDRAM	128		TN LCD HD	1366 x 768	23
21	Asus	76700	AMD	3.7	Ryzen 5	4	4	8	DDR4-SDRAM	512		IPS LCD Full HD	1920 x 1080	26
22	Dell	77700	Intel	4.2	Core i5 11th gen	4	8	8	DDR4-SDRAM	256		TN LCD Full HD	1920 x 1080	22
23	Lenovo	69900	AMD	3.7	Ryzen 5	4	4	8	DDR4-SDRAM	256		TN LCD Full HD	1920 x 1080	22
24	Lenovo	69900	AMD	3.7	Ryzen 3	4	2	8	DDR4-SDRAM	256		IPS LCD Full HD	1920 x 1080	22
25	Lenovo	105500	Intel	3.9	Core i7 10th gen	4	8	8	DDR4-SDRAM	512		IPS LCD Full HD	1920 x 1080	25

Figure 7. Dataset after removing redundancies

3.1.3 Assigning Numerical Values to Dataset

In order to do fuzzy clustering, I needed a fully numerical dataset so I studied certain aspects of specifications, like types of GPU featured in my dataset, and assigned satisfactory values accordingly. To give an example of what this process looked like I try to explain how I converted the brand name of a laptop into satisfactory values.

1. Determine the different classes in the column: Class("Brand") = {Microsoft, Lenovo, HP, Dell, Apple, Asus, MSI}

2. Research the success of each company. In this case I searched for company net worth online and ordered the companies from most successful to least successful. The ordered list is: Apple ($2.8 trillion), Microsoft ($1 trillion), Dell ($50.7 billion), HP ($35.81 billion), Lenovo ($14 billion), MSI ($5.23 billion), Asus ($1.3 billion).

3. Assign satisfactory values according to a ratio between target and max value, e.g., satisfactory value for Microsoft = $1 trillion / $2.8 trillion = 0.36. The results for this case are shown in table 2.

Table 2. Satisfactory values for different brands

Apple	**1.0000**
Microsoft	0.3571
Dell	0.0181
HP	0.0128
Lenovo	0.0050
MSI	0.0019
Asus	0.0005

Similarly, I constructed Table 3, Table 4, Table 5, Table 6, and Table 7. Note that the bigger the satisfactory value the better is the class.

Table 3. Satisfactory values for CPU manufecturers

AMD	**1**
Intel	0.43
Apple M1	0.12
ARM	0.08

Table 4. Satisfactory values for CPU type

Core i3 11th gen	**1**
Ryzen 9	0.919
M1	0.899
Ryzen 5	0.881
Ryzen 5	0.881
Core i5 10th gen	0.864
Pentium Gold	0.835
Core i3 10th gen	0.832
Athlon Gold	0.813
Core i7 11th gen	0.805
Ryzen 3	0.805
Core i5 11th gen	0.793
Microsoft SQ 1	0.788
Pentium Silver	0.764
Celeron	0.712
Ryzen 7	0.68
Core i7 10th gen	0.655

For deciding the satisfactory value of an OS certain rules were kept in mind: OS gains 5 points if it is free, OS gains 5 points if it is pleasant to use/look at, OS gains 4 points if it's game compatible, OS gains 3 points if it is widespread, and OS gains or looses up to 4 points judging from ease of use. Rankings are as follows: Windows 10 Pro = 0+5+4+3+2 = 14, Windows 10s = 0+5+4+0+2 = 11, Windows 10 Home = 0+5+4+3+2 = 14, macOS Big Sur = 5+5+0+3+3 = 16, Not installed = 0. Of course, the data was then normalized.

Table 5. Satisfactory values for choice of operating system (Windows 10 is the same as Windows 10 Home)

MacOS Big Sur	**1**
Windows 10 Pro	0.875
Windows 10 Home	0.875
Windows 10s	0.688
Not Installed	0

Table 6. Satisfactory values set to GPUs

Apple (8Core)	**1**
NVIDIA GeForce GTX 1650	0.353619
Apple (7Core)	0.281004
NVIDIA GeForce RTX 3070	0.178482
NVIDIA GeForce RTX 3060	0.178475
Nvidia GeForce RTX 2060	0.153257
AMD Radeon RX 5600	0.14963
NVIDIA GeForce GTX 1660	0.144222
NVIDIA GeForce GTX 1650	0.142419
Nvidia GeForce RTX 2070	0.120771
Intel Iris Plus Graphics	0.11351
Radeon Vega 8 Graphics	0.106364
Radeon RX Vega 10	0.099178
Intel Iris Xe Graphics	0.099153
Intel UHD Graphics	0.095547
AMD Radeon Graphics	0.095547
Intel UHD Graphics 605	0.091941
Adreno 685 GPU	0.025511

For classifying better display standards integers were user, from 1 to 8 as in Table 7 below, with 8 being the better choice.

Table 7. Satisfactory values for classifying display technologies

PixelSense	**8**
Retina Display	7
IPS	6
SVA	5
TN	4
LED AntiGlare Full HD	3
LCD HD	2
Full HD	1

3.1.3. Summary: How did we get to numerical data?

In total, I used 3 techniques to manipulate data: assigning satisfactory values to non-numerical data, normalizing the numerical data, and preserving form while changing widespread/range of data. The satisfactory value method is explained in 3.1.2., it deals with comparison between choices for a feature. The end result is a value between 0 and 1 included representing satisfaction with our choice. Sometimes an integer range is used for simplicity (Table 7).

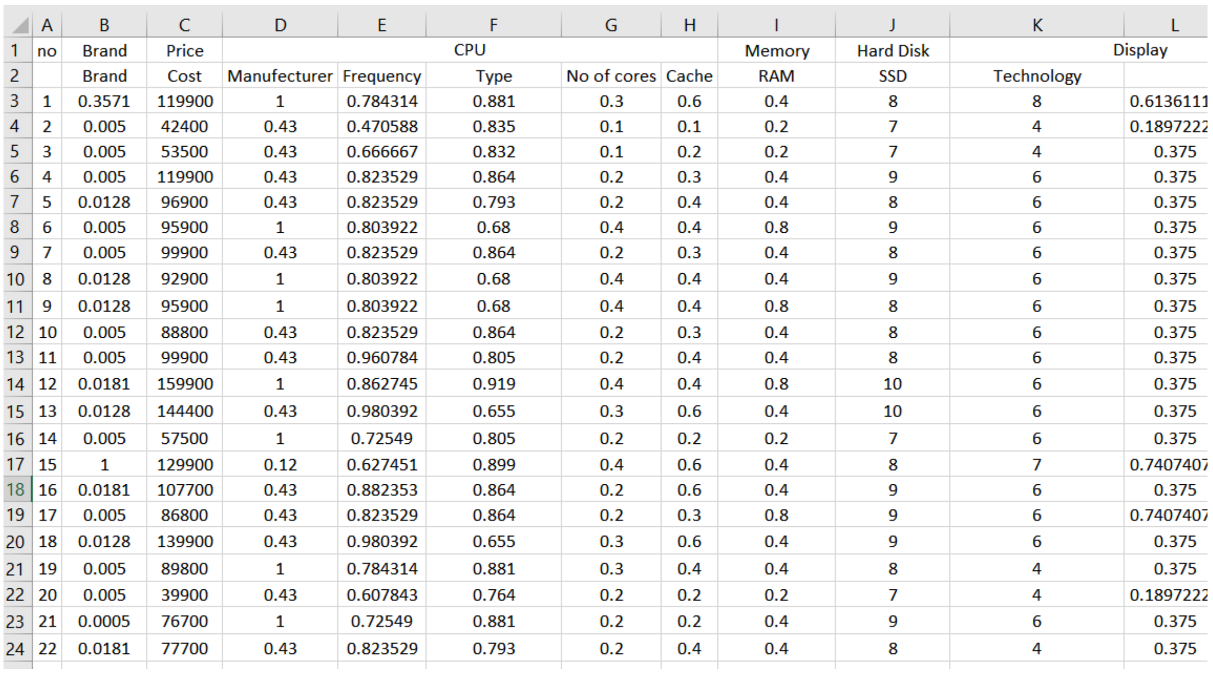
The other method, normalizing the numerical data, has as its aim to preserve the overall structure by shrinking or enlarging the data space. This is done in two steps:

1. Find the maximum in the colon.

2. Divide each colon entry by the maximum.

It results in data between the range 0 – 1 that is easier to digest when dealing with graphs that we later will.

The last, but not least, approach was used as a logical conclusion to some types of numerical data. For example, the storage capacity of the SSD is a power of 2, so it would be best to get the logarithm of base 2 of the whole column, so we will not have to show large values or confusing data as graphs later. All this work concludes a fully numerical data (Figure 8) ready to be used. Some tips and tricks like getting rid of a hierarchical division of data and exporting the dataset into a csv file help us greatly implement our solution on the next step.



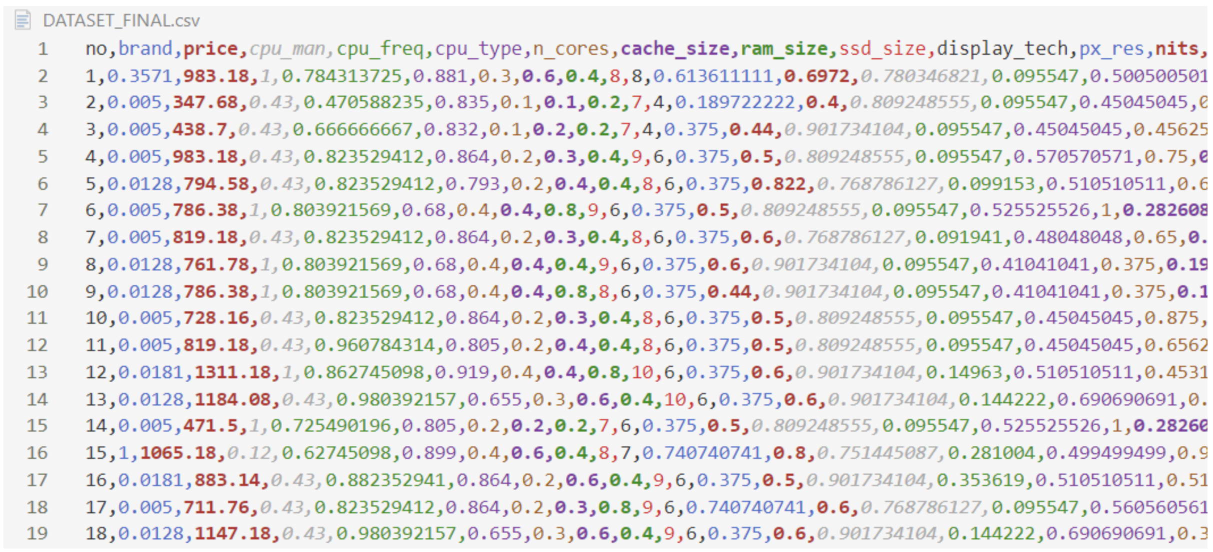
	A	B	C	D	E	F	G	H	I	J	K	L
1	no	Brand	Price			CPU			Memory	Hard Disk		Display
2		Brand	Cost	Manufecturer	Frequency	Type	No of cores	Cache	RAM	SSD	Technology	
3	1	0.3571	119900	1	0.784314	0.881	0.3	0.6	0.4	8	8	0.6136111
4	2	0.005	42400	0.43	0.470588	0.835	0.1	0.1	0.2	7	4	0.1897222
5	3	0.005	53500	0.43	0.666667	0.832	0.1	0.2	0.2	7	4	0.375
6	4	0.005	119900	0.43	0.823529	0.864	0.2	0.3	0.4	9	6	0.375
7	5	0.0128	96900	0.43	0.823529	0.793	0.2	0.4	0.4	8	6	0.375
8	6	0.005	95900	1	0.803922	0.68	0.4	0.4	0.8	9	6	0.375
9	7	0.005	99900	0.43	0.823529	0.864	0.2	0.3	0.4	8	6	0.375
10	8	0.0128	92900	1	0.803922	0.68	0.4	0.4	0.4	9	6	0.375
11	9	0.0128	95900	1	0.803922	0.68	0.4	0.4	0.8	8	6	0.375
12	10	0.005	88800	0.43	0.823529	0.864	0.2	0.3	0.4	8	6	0.375
13	11	0.005	99900	0.43	0.960784	0.805	0.2	0.4	0.4	8	6	0.375
14	12	0.0181	159900	1	0.862745	0.919	0.4	0.4	0.8	10	6	0.375
15	13	0.0128	144400	0.43	0.980392	0.655	0.3	0.6	0.4	10	6	0.375
16	14	0.005	57500	1	0.72549	0.805	0.2	0.2	0.2	7	6	0.375
17	15	1	129900	0.12	0.627451	0.899	0.4	0.6	0.4	8	7	0.7407407
18	16	0.0181	107700	0.43	0.882353	0.864	0.2	0.6	0.4	9	6	0.375
19	17	0.005	86800	0.43	0.823529	0.864	0.2	0.3	0.8	9	6	0.7407407
20	18	0.0128	139900	0.43	0.980392	0.655	0.3	0.6	0.4	9	6	0.375
21	19	0.005	89800	1	0.784314	0.881	0.3	0.4	0.4	8	4	0.375
22	20	0.005	39900	0.43	0.607843	0.764	0.2	0.2	0.2	7	4	0.1897222
23	21	0.0005	76700	1	0.72549	0.881	0.2	0.2	0.4	9	6	0.375
24	22	0.0181	77700	0.43	0.823529	0.793	0.2	0.4	0.4	8	4	0.375

Figure 8. Fully numerical data in Excel

In Figure 9, the naming of features is changed as it will be used in Python. The final description of data would be as follows: 70 laptop computers have their features separated in brand name, price, CPU manufacturer, CPU frequency, CPU type, number of CPU cores, size of L2 cache, RAM size, SSD Size, display technology, resolution, brightnesss (in nits), width of screen, graphics card, battery capacity, battery durability, AC adapter power, operating system, and fingerprint security. The data is fully numerical, with most columns normalized to be between 0 and 1. The price, that will be our target for comparison, must be in Euro so it is changed accordingly (from LEK to EUR, as of 01 July 2021).

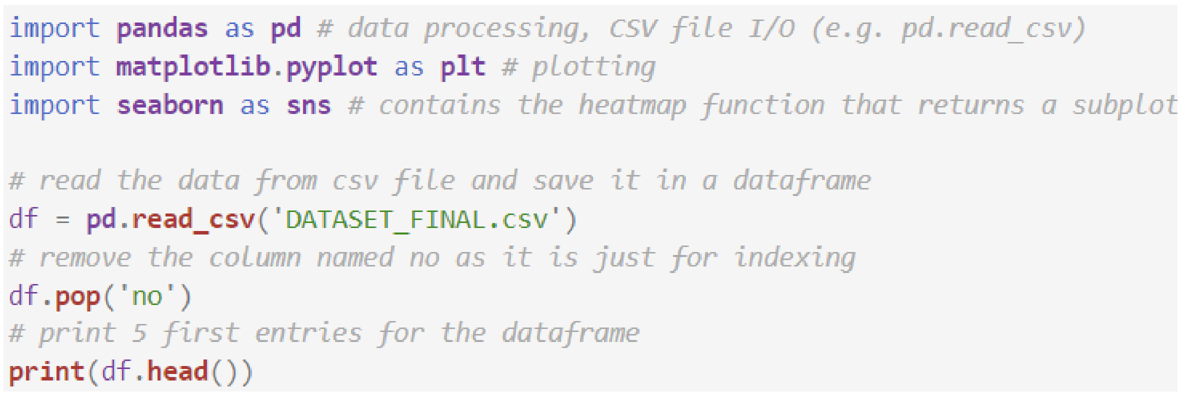
3.2. Implementation

For this project I used Visual Studio Code IDE, which is a very flexible IDE that has all the necessary tools to handle the coding. For better management of data, the Excel database was exported in a csv file called DATASET_FINAL.csv (Figure 9) as loading dataframes from csv files can ve done really easily in Python using the pandas module (Figure 10). In Figure 9, color is used to differentiate features, for example, values 0.6972, 0.4, etc., in hard red belong to the feature *nits*.



```
DATASET_FINAL.csv
no,brand,price,cpu_man,cpu_freq,cpu_type,n_cores,cache_size,ram_size,ssd_size,display_tech,px_res,nits,
1,0.3571,983.18,1,0.784313725,0.881,0.3,0.6,0.4,8,8,0.613611111,0.6972,0.780346821,0.095547,0.500500501
2,0.005,347.68,0.43,0.470588235,0.835,0.1,0.1,0.2,7,4,0.189722222,0.4,0.809248555,0.095547,0.45045045,0
3,0.005,438.7,0.43,0.666666667,0.832,0.1,0.2,0.2,7,4,0.375,0.44,0.901734104,0.095547,0.45045045,0.45625
4,0.005,983.18,0.43,0.823529412,0.864,0.2,0.3,0.4,9,6,0.375,0.5,0.809248555,0.095547,0.570570571,0.75,0
5,0.0128,794.58,0.43,0.823529412,0.793,0.2,0.4,0.4,8,6,0.375,0.822,0.768786127,0.099153,0.510510511,0.6
6,0.005,786.38,1,0.803921569,0.68,0.4,0.4,0.8,9,6,0.375,0.5,0.809248555,0.095547,0.525525526,1,0.282608
7,0.005,819.18,0.43,0.823529412,0.864,0.2,0.3,0.4,8,6,0.375,0.6,0.768786127,0.091941,0.48048048,0.65,0.
8,0.0128,761.78,1,0.803921569,0.68,0.4,0.4,0.4,9,6,0.375,0.6,0.901734104,0.095547,0.41041041,0.375,0.19
9,0.0128,786.38,1,0.803921569,0.68,0.4,0.4,0.8,8,6,0.375,0.44,0.901734104,0.095547,0.41041041,0.375,0.1
10,0.005,728.16,0.43,0.823529412,0.864,0.2,0.3,0.4,8,6,0.375,0.5,0.809248555,0.095547,0.45045045,0.875,
11,0.005,819.18,0.43,0.960784314,0.805,0.2,0.4,0.4,8,6,0.375,0.5,0.809248555,0.095547,0.45045045,0.6562
12,0.0181,1311.18,1,0.862745098,0.919,0.4,0.4,0.8,10,6,0.375,0.6,0.901734104,0.14963,0.510510511,0.4531
13,0.0128,1184.08,0.43,0.980392157,0.655,0.3,0.6,0.4,10,6,0.375,0.6,0.901734104,0.144222,0.690690691,0.
14,0.005,471.5,1,0.725490196,0.805,0.2,0.2,0.2,7,6,0.375,0.5,0.809248555,0.095547,0.525525526,1,0.28260
15,1,1065.18,0.12,0.62745098,0.899,0.4,0.6,0.4,8,7,0.740740741,0.8,0.751445087,0.281004,0.499499499,0.9
16,0.0181,883.14,0.43,0.882352941,0.864,0.2,0.6,0.4,9,6,0.375,0.5,0.901734104,0.353619,0.510510511,0.51
17,0.005,711.76,0.43,0.823529412,0.864,0.2,0.3,0.8,9,6,0.740740741,0.6,0.768786127,0.095547,0.560560561
18,0.0128,1147.18,0.43,0.980392157,0.655,0.3,0.6,0.4,9,6,0.375,0.6,0.901734104,0.144222,0.690690691,0.3
```

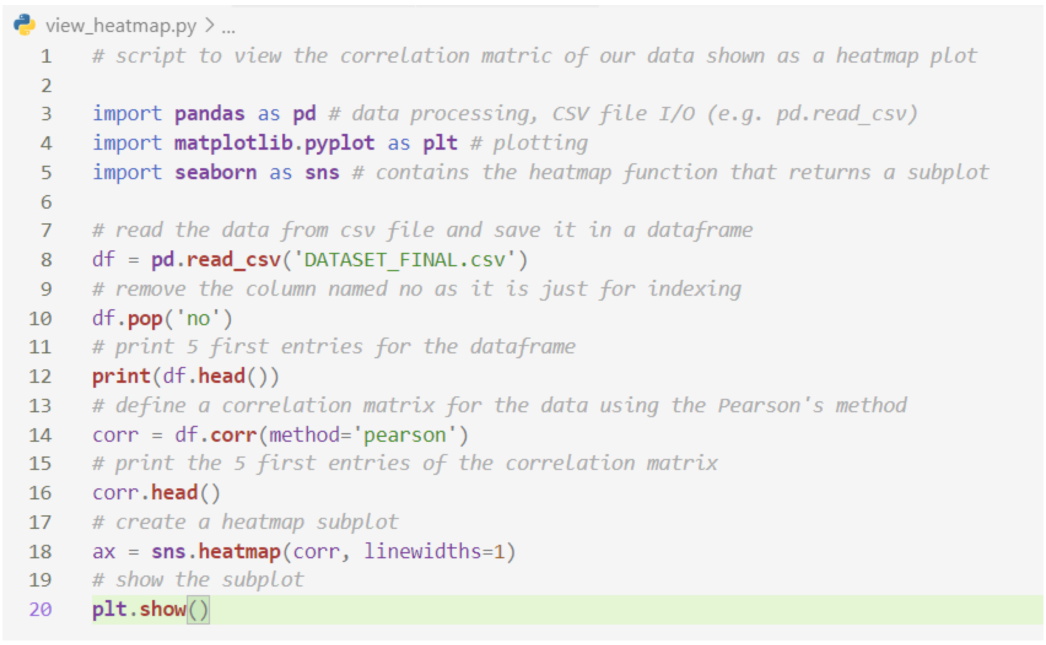
Figure 9. DATASET_FINAL.csv file



```python
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # plotting
import seaborn as sns # contains the heatmap function that returns a subplot

# read the data from csv file and save it in a dataframe
df = pd.read_csv('DATASET_FINAL.csv')
# remove the column named no as it is just for indexing
df.pop('no')
# print 5 first entries for the dataframe
print(df.head())
```

Figure 10. Manipulating and showing data with pandas module

To give some perspective on data, I put together a small script in view_heatmap.py file to present a heatmap for the correlation matrix between features, which shows what features are more dependent on other features with lighter shades of red showing more dependancy. Below you can find the script (Figure 11) and its output plot (Figure 12Figure 11).



```python
# script to view the correlation matric of our data shown as a heatmap plot

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # plotting
import seaborn as sns # contains the heatmap function that returns a subplot

# read the data from csv file and save it in a dataframe
df = pd.read_csv('DATASET_FINAL.csv')
# remove the column named no as it is just for indexing
df.pop('no')
# print 5 first entries for the dataframe
print(df.head())
# define a correlation matrix for the data using the Pearson's method
corr = df.corr(method='pearson')
# print the 5 first entries of the correlation matrix
corr.head()
# create a heatmap subplot
ax = sns.heatmap(corr, linewidths=1)
# show the subplot
plt.show()
```

Figure 11. Script to show heatmap of correlation matrix

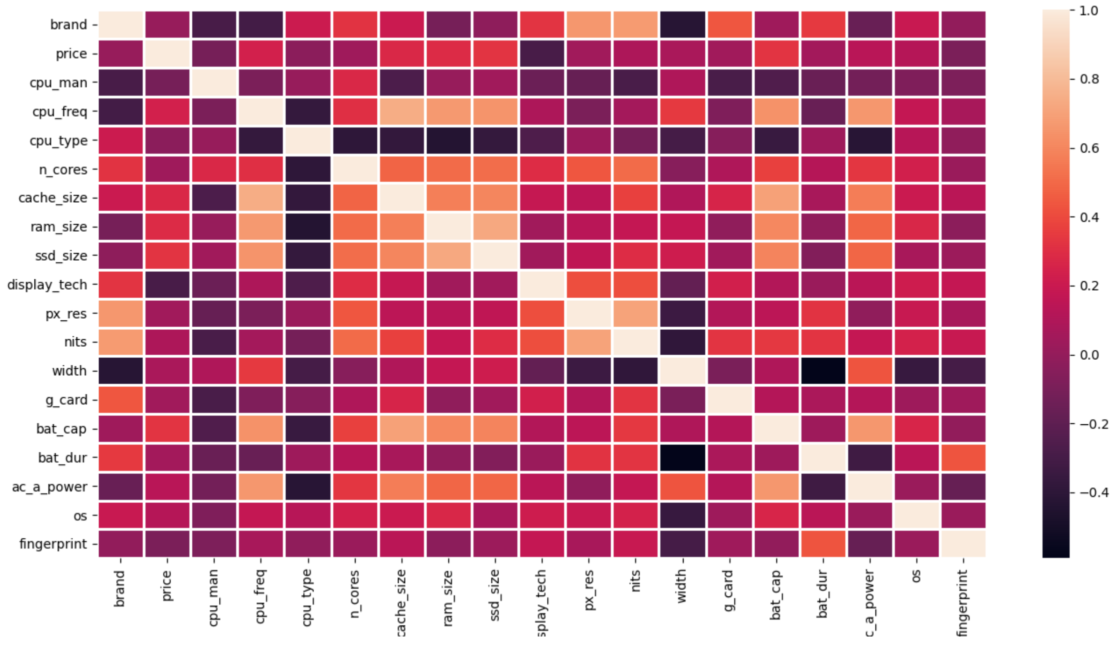
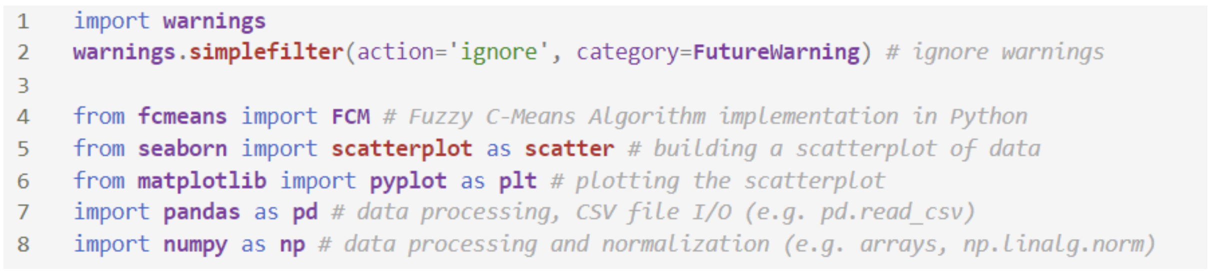


Figure 12. Correlation matrix heatmap of features

3.3. Result Graph Building

To choose the best features that influence the price mostly, two Python scripts were put together. They have similar structure except for one of them computes 2 fuzzy clusters and the other computes 3 on our data, so here we explain the script that divides the data that we feed into it into 2 clusters. I named the files graph_with_2.py and graph_with_3.py respectively. Code shown below is from graph_with_2.py file.

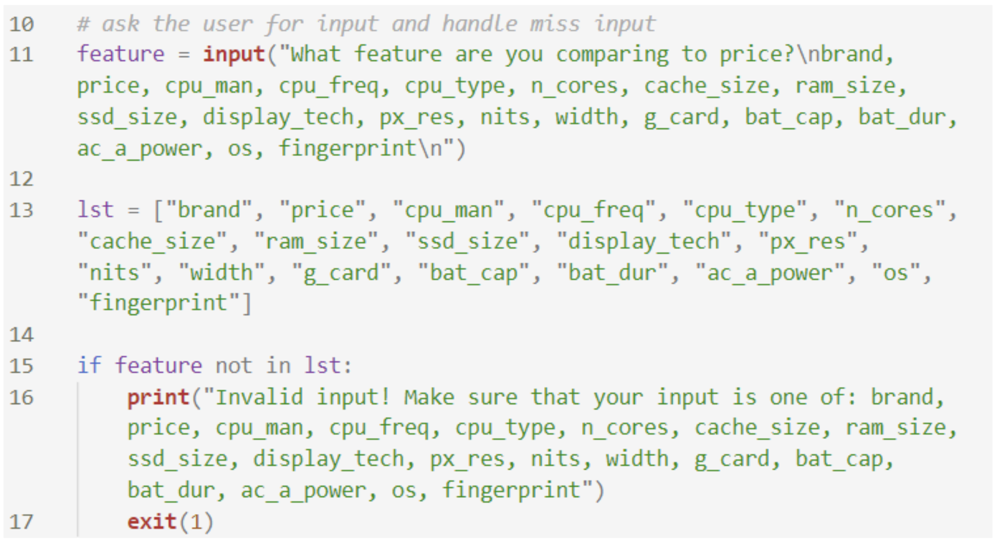
Firstly, we import all the necessary python modules. Figure 13 shows the lines of code that handle the imports together with a simple description of what do we use them for. Lines 1 and 2 are there to suppress future warnings, which are warnings for functions set to change in the future releases of a module.



```
1   import warnings
2   warnings.simplefilter(action='ignore', category=FutureWarning) # ignore warnings
3
4   from fcmeans import FCM # Fuzzy C-Means Algorithm implementation in Python
5   from seaborn import scatterplot as scatter # building a scatterplot of data
6   from matplotlib import pyplot as plt # plotting the scatterplot
7   import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
8   import numpy as np # data processing and normalization (e.g. arrays, np.linalg.norm)
```

Figure 13. Importing the necessary Python modules

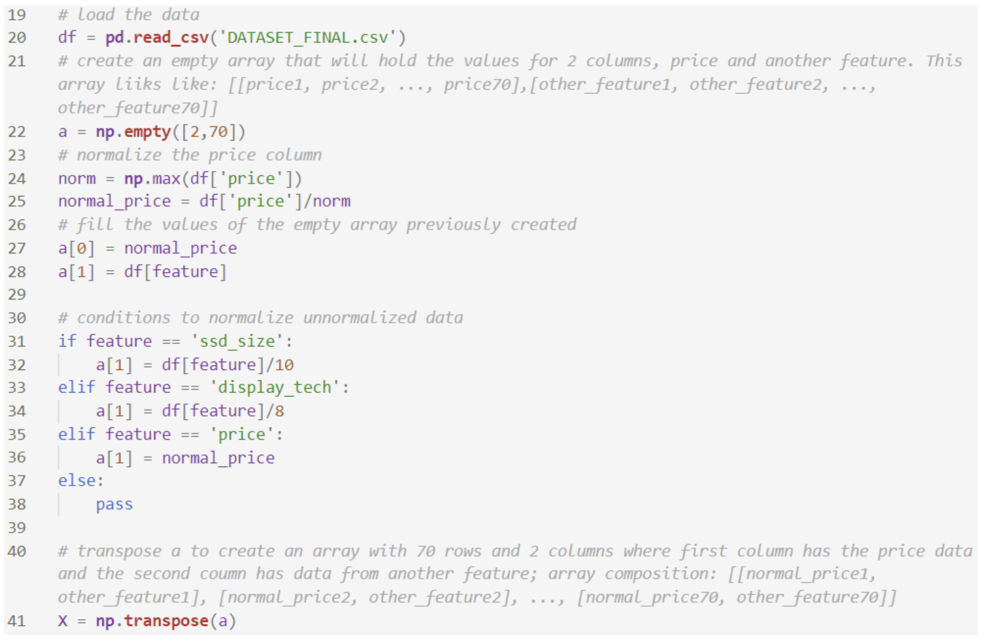
Next, we validate input. Our input can be any of the 19 features in a shortened form. If input is invalid, we exit the program without going further. The lines of code handling input validation are shown below (Figure 14):



```python
# ask the user for input and handle miss input
feature = input("What feature are you comparing to price?\nbrand, price, cpu_man, cpu_freq, cpu_type, n_cores, cache_size, ram_size, ssd_size, display_tech, px_res, nits, width, g_card, bat_cap, bat_dur, ac_a_power, os, fingerprint\n")

lst = ["brand", "price", "cpu_man", "cpu_freq", "cpu_type", "n_cores", "cache_size", "ram_size", "ssd_size", "display_tech", "px_res", "nits", "width", "g_card", "bat_cap", "bat_dur", "ac_a_power", "os", "fingerprint"]

if feature not in lst:
    print("Invalid input! Make sure that your input is one of: brand, price, cpu_man, cpu_freq, cpu_type, n_cores, cache_size, ram_size, ssd_size, display_tech, px_res, nits, width, g_card, bat_cap, bat_dur, ac_a_power, os, fingerprint")
    exit(1)
```

Figure 14. Input and input validation

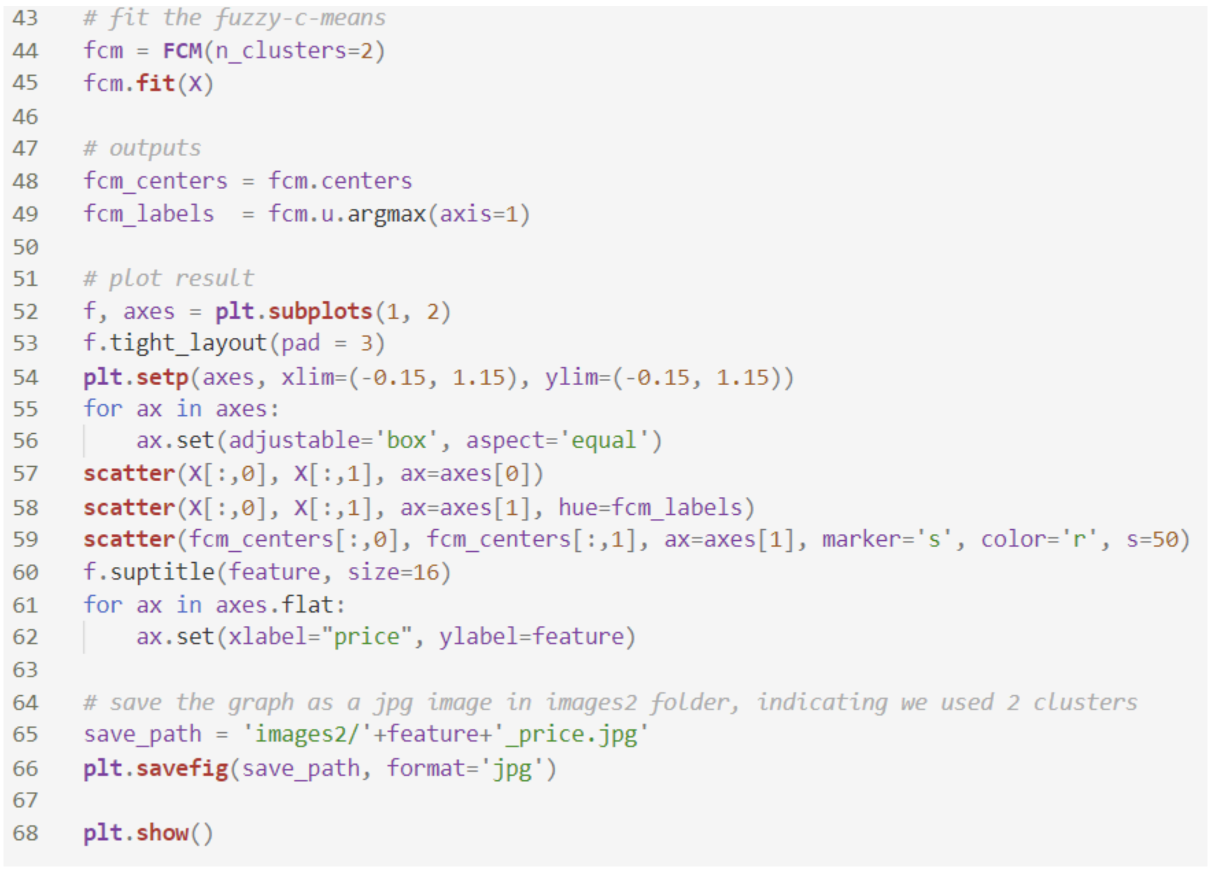
What is left is to load the data and do the necessary modifications to it. Loading the data is relatively easy with pandas module (only one line of code). We save 2 arrays in our data, price and some other feature, into a numpy array that has extended tools to deal with statistical data. Using numpy, we normalize the price data and save the final form of our array in array X. A more detailed explanation is provided through comments in Figure 15 showing the Python code to load data.



```python
# load the data
df = pd.read_csv('DATASET_FINAL.csv')
# create an empty array that will hold the values for 2 columns, price and another feature. This array liiks like: [[price1, price2, ..., price70],[other_feature1, other_feature2, ..., other_feature70]]
a = np.empty([2,70])
# normalize the price column
norm = np.max(df['price'])
normal_price = df['price']/norm
# fill the values of the empty array previously created
a[0] = normal_price
a[1] = df[feature]

# conditions to normalize unnormalized data
if feature == 'ssd_size':
    a[1] = df[feature]/10
elif feature == 'display_tech':
    a[1] = df[feature]/8
elif feature == 'price':
    a[1] = normal_price
else:
    pass

# transpose a to create an array with 70 rows and 2 columns where first column has the price data and the second coumn has data from another feature; array composition: [[normal_price1, other_feature1], [normal_price2, other_feature2], ..., [normal_price70, other_feature70]]
X = np.transpose(a)
```

Figure 15. Loading data and preparing it for Fuzzy C-Means

The next portion of the code deals with plotting a graph between price (x-axis) and another feature (y-axis), showing raw data and the clusters formed by FCM, and showing the centroids of the clusters. Code differs somewhat from graph_with_3.py file but the logic is the same. Below you can see the code (Figure 16) and an example graph between price and cpu_freq (Figure 17).



```python
# fit the fuzzy-c-means
fcm = FCM(n_clusters=2)
fcm.fit(X)

# outputs
fcm_centers = fcm.centers
fcm_labels  = fcm.u.argmax(axis=1)

# plot result
f, axes = plt.subplots(1, 2)
f.tight_layout(pad = 3)
plt.setp(axes, xlim=(-0.15, 1.15), ylim=(-0.15, 1.15))
for ax in axes:
    ax.set(adjustable='box', aspect='equal')
scatter(X[:,0], X[:,1], ax=axes[0])
scatter(X[:,0], X[:,1], ax=axes[1], hue=fcm_labels)
scatter(fcm_centers[:,0], fcm_centers[:,1], ax=axes[1], marker='s', color='r', s=50)
f.suptitle(feature, size=16)
for ax in axes.flat:
    ax.set(xlabel="price", ylabel=feature)

# save the graph as a jpg image in images2 folder, indicating we used 2 clusters
save_path = 'images2/'+feature+'_price.jpg'
plt.savefig(save_path, format='jpg')

plt.show()
```

Figure 16. Code handling graph building and plotting

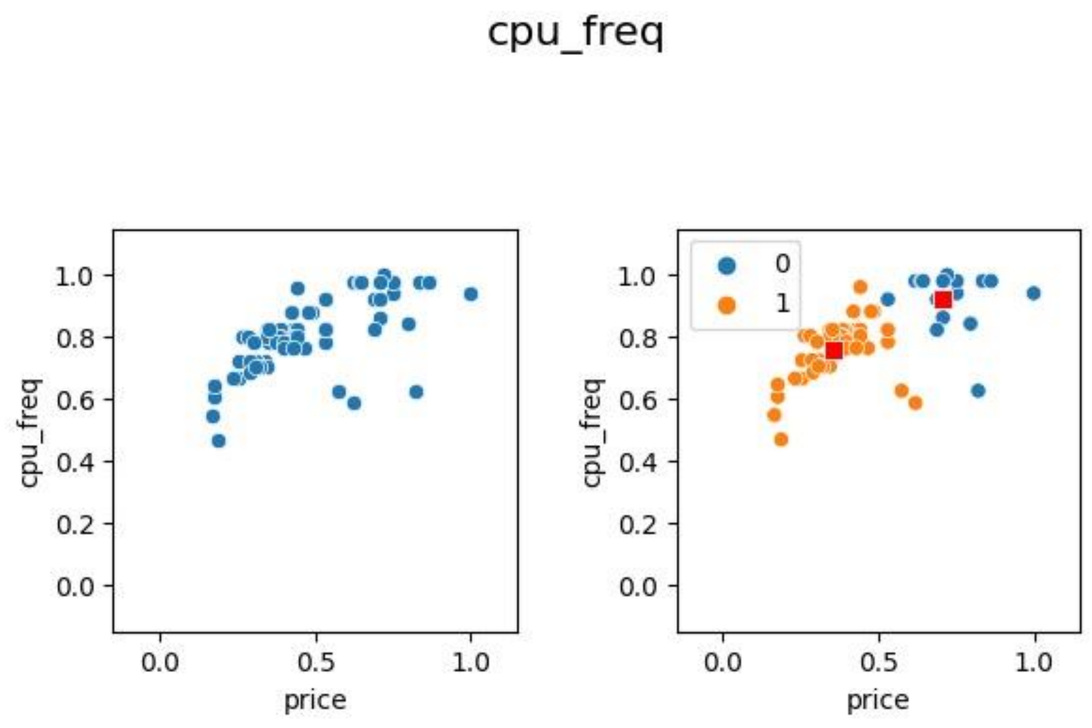


Figure 17. Price vs. CPU frequency: raw data on the left and Fuzzy C-Means clustering (c=2) on the right

To make graph plotting easier, I modified the code in graph_with_2.py to create automated2.py and graph_with_3.py to create automated3.py, and automated the whole process of graph creation and saving. To view all 76 graphs, go to APPENDIX B.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1. Graph Interpretation

A graph is divided into 2 subplots: on the left the raw data mapping of the points is shown, on the right the Fuzzy C-Means clustering is plotted with 2 or 3 clusters. Let us take Figure 17 as a reference; the legend of the graph on the right is meant to simply show the division of clusters. What interests us in these graphs are summarized in these 3 observations: how closely the centroids of the clusters describe a line that has a slope of 1 or -1, how similar are the clusters with each other, and how these observations change when we move from 2 clusters to 3 clusters. If the slope of the line defined by cluster centroids is 1 then the price is very dependent on the feature in a positive way (if the features “increases”, the price increases), if the slope is -1 than the feature is correlated negatively (if the features “increases”, the price decreases), and if the slope is not close to 1 or -1 a change in feature has little to no effect on price. Cluster similarity is a measure of fair distribution of data: in Figure 17 clusters are fairly similar, this means that price and CPU frequency relation can be described mostly by the first observation. In our case, the fact that the clusters are similar justifies our conclusion from this graph that is “Price is positively and heavily dependent on the CPU frequency as the centroids of the 2 clusters describe a line that has a slope close to 1”. To explore these observations and the conclusions we get from them further, I choose 4 test samples: price on price, CPU frequency on price, battery durability on price, and fingerprint reader on price.

4.1.1. Price on Price: the Perfect Correlation

In this case, we want to conclude how the price affects the price. It is a rather abstract point of view, but it gives great insight in how a perfect positive correlation between the price and a feature would look like. Figure 18 shows how all the points lay on the x=y line, and of course, the centroids of the clusters do lay on the x=y line too, which has a slope of 1. Clusters are also very similar in shape.

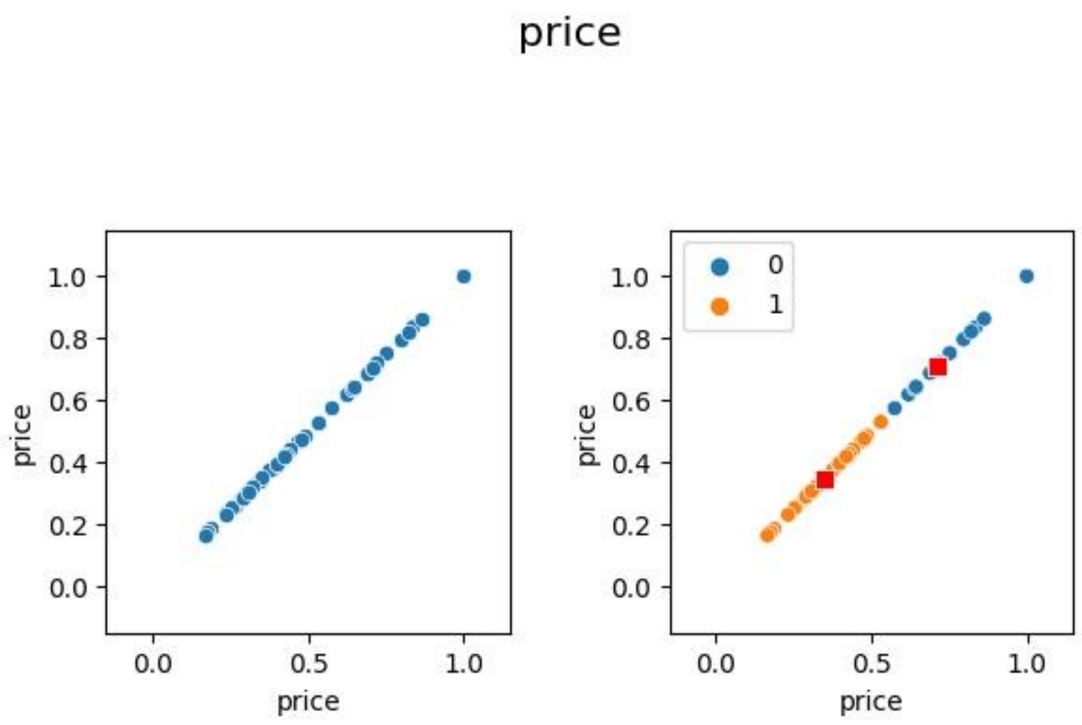


Figure 18. Price vs. price, the perfect positive correlation

4.1.2. CPU Frequency on Price: A Good Correlation

This case was previously discussed in section 4.1., and our conclusion was that CPU frequency strongly affects the price in a positive way. To support this statement, we can go a step further and take a look at Figure 19 which shows the same graph as Figure 17, but this time data is separated between 3 clusters. The centroids of these clusters (shown as black x) can be connected into lines creating 3 lines, and if we were to find the mean line slope (by adding each slope of these lines and dividing by 3) for these 3 lines we would have a result close to 1.

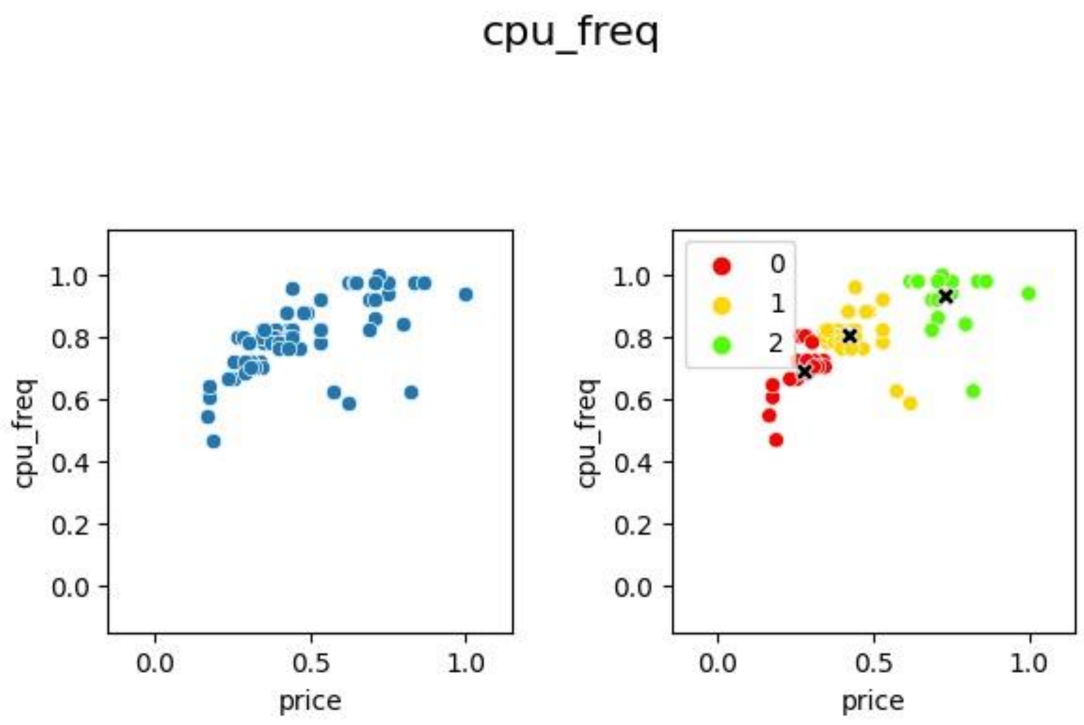


Figure 19. FCM with 3 clusters in price vs CPU frequency graph

4.1.3. Battery Durability on Price: the Tricky Test Case

A first look on the graph for this test case gives all the positive signs to conclude that battery durability has a strong and positive effect on price (Figure 20), but when we dig deeper, we can see that the correlation between these features is not what it seemed. The centroids of the clusters when we use 3 clusters for FCM form a triangle that has 3 acute angles (Figure 21), meaning that battery durability has limited effect on price. Although, battery choice affects price positively, it does not affect it strongly. The conclusion is that battery durability affects price lightly, in a positive way.

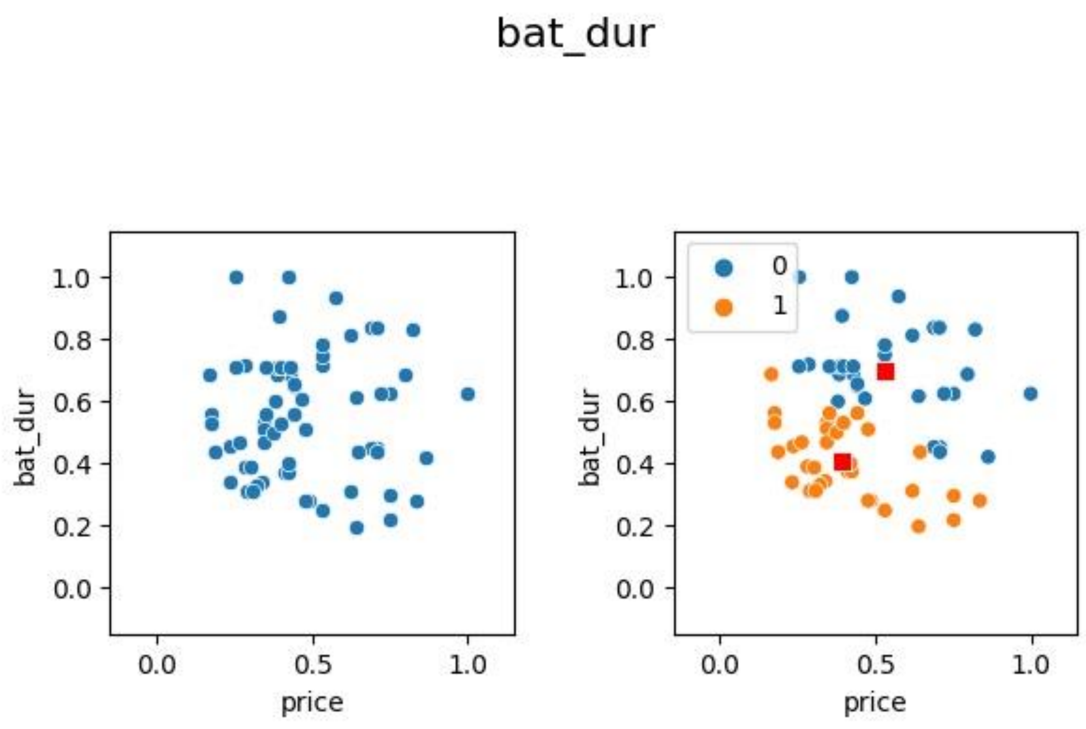


Figure 20. Battery durability effect on price as viewed with 2 clusters

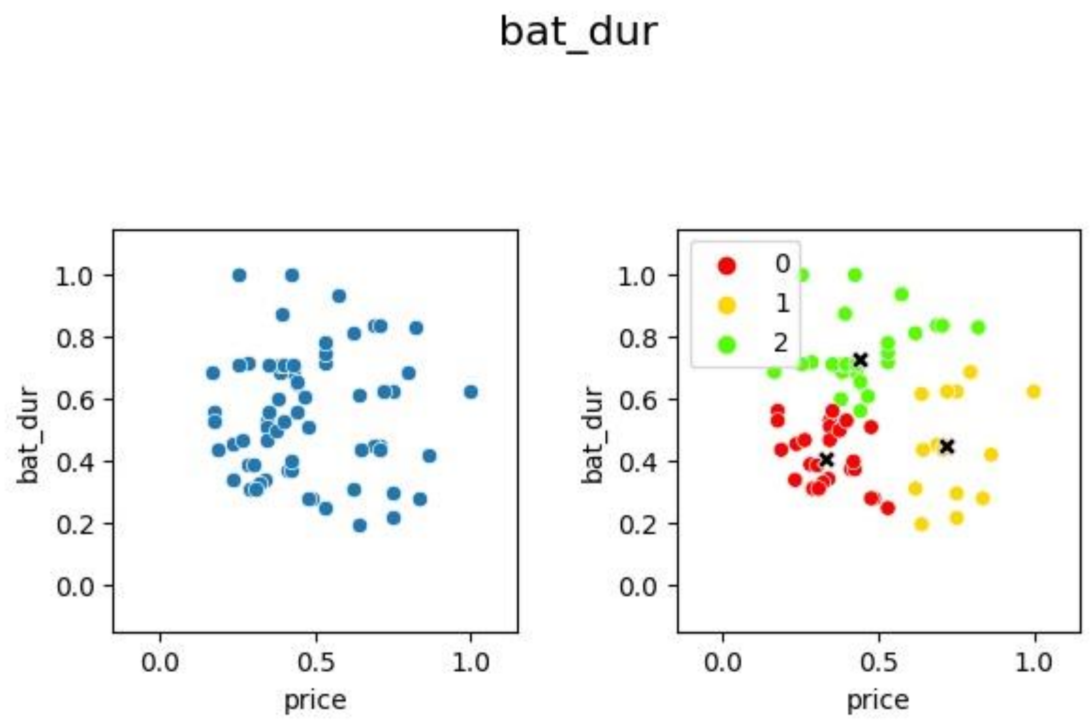


Figure 21. Battery durability effect on price as viewed with 3 clusters

4.1.4. Fingerprint Reader on Price: the Bad Correlation

No much words to say here, just wanted to show a case of bad correlation. The absence or presence of a fingerprint reader has slim to none effect on the price of a laptop as the centroids of the clusters form a line that has a very large (approaching infinity) slope. This statement is backed up by the similarity of the clusters and by the fact that FCM with 3 clusters produces centroids that create an acute-angled triangle.

Figure 22 and Figure 23 show the respective graphs that made us arrive at this conclusion.

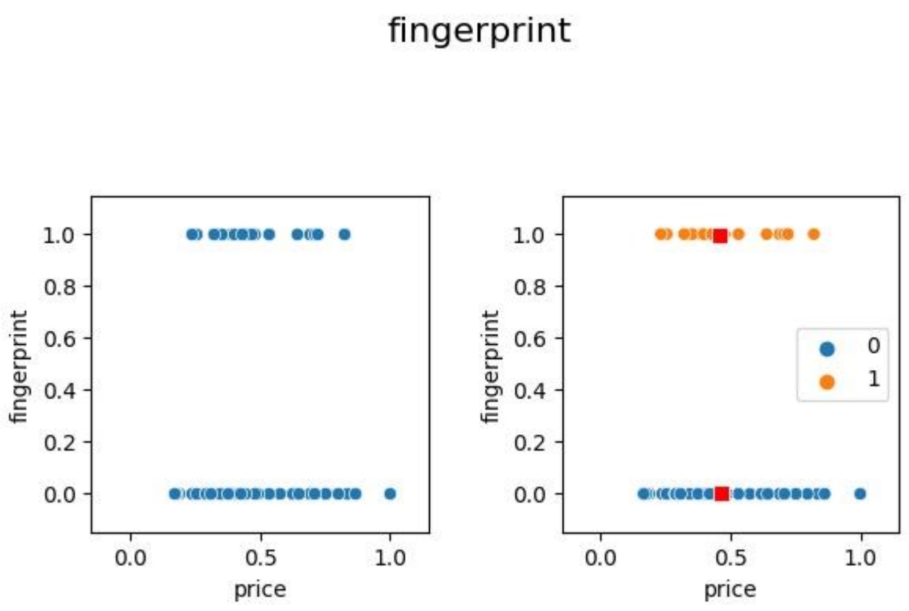


Figure 22. Fingerprint reader on price: FCM with 2 clusters

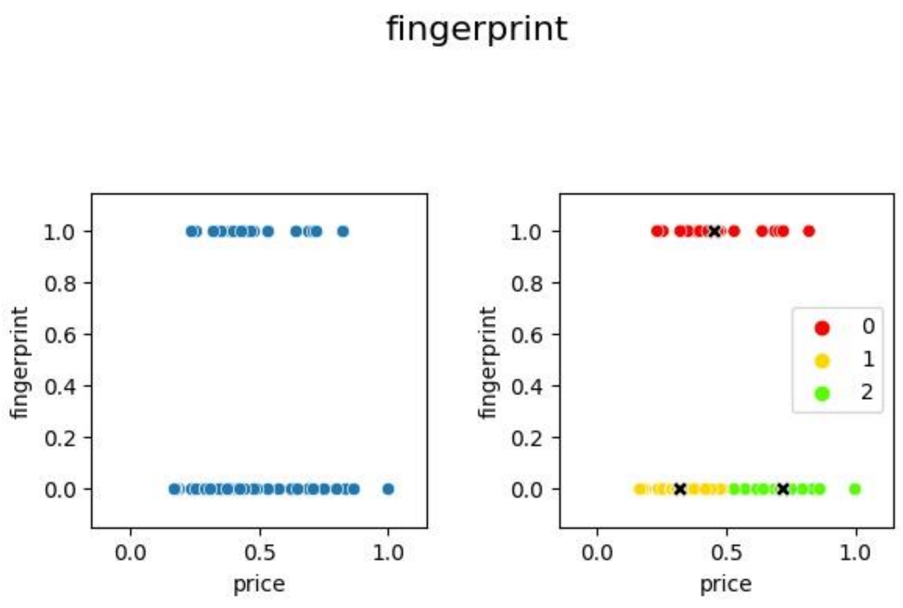


Figure 23. Fingerprint reader on price: FCM with 3 clusters

4.2. Final Results

To facilitate the process of measuring how much a feature affects the price, I modified my code in graph_with_2.py and graph_with_3.py files to show regression line between centroids and lines x=y, x=-y. I named these files automatedR2.py and automatedR3.py respectively. The source code of each file mentioned throughout the text and more can be found in APPENDIX A. Two modified graphs can be seen in Figure 24 and Figure 25below. Graph in Figure 24 shows how price is strongly, positively correlated with RAM size by computing 2 cluster centroids and the regression line between them. Figure 25 does the same thing but this time with 3 cluster centroids.

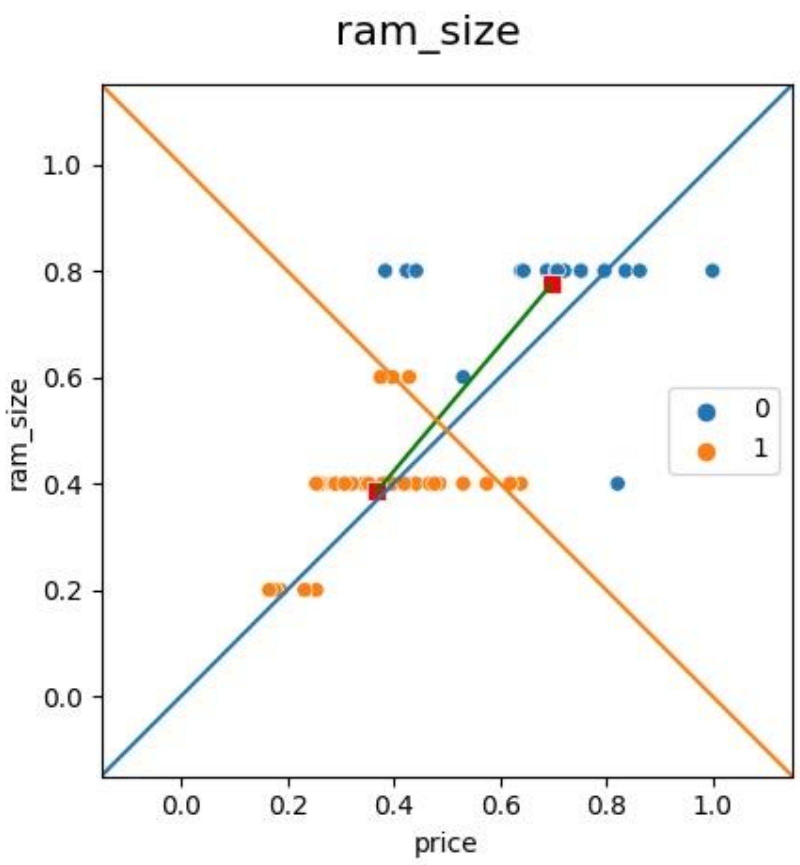


Figure 24. Regression line between cluster centroids (dark green), x=y line (blue), x=-y line (orange) on a graph between RAM size and price divided in 2 clusters

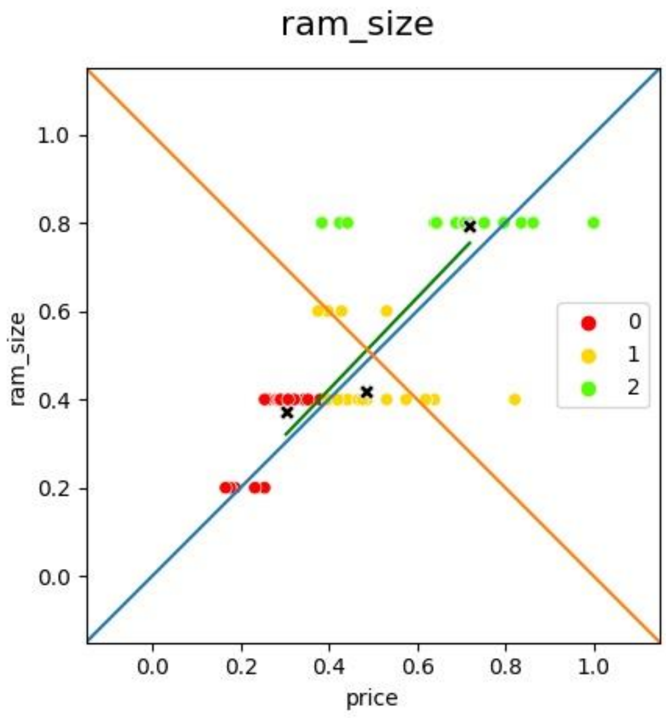


Figure 25. Regression line between cluster centroids (dark green), x=y line (blue), x=-y line (orange) on a graph between RAM size and price divided in 3 clusters

After careful study of these graphs, I have deduced that:

1. Price is strongly and positively dependent on RAM size, battery capacity, L2 cache size, AC adapter power, display technology used, brightness of display (nits), CPU frequency, and solid-state disk size.

2. Price is lightly but positively dependent on number of CPU cores, the GPU, and the brand of the laptop.

3. Price is not dependent or is little dependent on the resolution of the screen, battery durability (hours), type of OS installed, whether the laptop has a fingerprint reader or not, and the width of the screen.

4. Suprisingly, price is negatively dependent on CPU type (i3, i5, Ryzen, etc.) and CPU manufacturer. This may be due to bad assumptions and conduct of research made before. In my defence, it is rather hard to find a good measure for CPU comparison.

Note that the feature effect on price is in decreasing order above.

CHAPTER 5

CONCLUSION

In conclusion, this paper has as an objective to do a market analysis of laptop market in Albania, and does that by using fuzzy logic and fuzzy c-means clustering algorithm. Through graphs plotted with Python programming language, I was able to show and analyze the data of 70 different laptops that I gathered myself. 18 features were compared and analyzed to derive a list of these features ordered from the most effecting feature to the least effecting one in comparison to the effect that such features had on price. This paper found that price is strongly and positively dependent on RAM size, battery capacity, L2 cache size, AC adapter power, display technology used, brightness of display (nits), CPU frequency, and solid-state disk size. Conclusions for light depndancy, no dependency and negative dependency are stated as well (*See 4.2*).

REFERENCES

[1] L. A. Zadeh, "Fuzzy logic.," *Computer,* vol. 21, no. 4, pp. 83-93, 1988.

[2] L. A. Zadeh, "Fuzzy logic: issues, contentions and perspectives.," *Proceedings of ICASSP'94. IEEE International Conference on Acoustics, Speech and Signal Processing.,* vol. 6, 1994.

[3] G. H. Ball and D. J. Hall., "A clustering technique for summarizing multivariate data.," *Behavioral science,* vol. 12, no. 2, pp. 153-155, 1967.

[4] "WolframAlpha," 2004. [Online]. Available: https://reference.wolfram.com/legacy/applications/fuzzylogic/Manual/12.html. [Accessed 1 July 2021].

[5] Y. C. Chen, Y. H. Chiu, C. W. Huang and C. H. Tu, "The analysis of bank business performance and market risk—Applying Fuzzy DEA.," *Economic modelling,* vol. 32, pp. 225-232, 2013.

[6] B. M. Balachandran and M. Mohammadian, "Development of a fuzzy-based multi-agent system for e-commerce settings.," *Procedia Computer Science,* vol. 60, pp. 593-602, 2015.

[7] A. P. U. Siahaan, "Decision Support System in Selecting The Appropriate Laptop Using Simple Additive Weighting.," 2017.

APPENDIX A

Code Implementation and Simple Explanation

File name: extract.py

Purpose: go over a list of URLs and extract information that is to be saved in data.json file

```
import urllib.request       # to access the link
import re       # regular expression support
# open source file
with open("raw.txt", 'r') as file:
    data = file.read()
lst = []
for i in data.split('href="'):
    lst.append(i.split('"')[0])
lst.pop(0)
def create_d(j):
    fp = urllib.request.urlopen(j)
    mybytes = fp.read()
    mystr = mybytes.decode("utf8")
    fp.close()
    val = re.findall(r"<p>.*?</p>", mystr)
    key = re.findall(r'<div class="font-bold uppercase text-sm text-gray-
800 woocommerce-product-attributes-item__label">.*?</div>', mystr)
    pure_keys=[]
    pure_val=[]
    for i in key:
        if i[i.find(">")+1:i[1:].find("<")+1] == "Details":
            continue
        else:
            pure_keys.append(i[i.find(">")+1:i[1:].find("<")+1])
    for j in val:
        pure_val.append(j[j.find(">")+1:j[1:].find("<")+1])
    d = {}
    for i in range(len(pure_keys)):
        if i<len(pure_val):
            d[pure_keys[i]] = pure_val[i]
    return d
# a list of dictionaries with all info
TO_BE_SAVED_LIST = []
for link in lst:
    TO_BE_SAVED_LIST.append({link: create_d(link)})
import pickle
with open("saved.bin", "wb")as file:
    pickle.dump(TO_BE_SAVED_LIST, file)
```

File name: data.json

Purpose: store laptop information for quick retrieval. Only one entry shown below, more than 111 entries originally. (This file is automatically created by extract.py)

```
{
    "https://shop.shpresa.al/product/lenovo-ideapad-s340-15iil-15-6-8gb-256gb-ssd-core-i7/": {
      "Pesh\u00eb": "",
      "Brand": "",
      "Color": "Dark Blue",
      "Type": "Ideapad",
      "Processor": "Intel\u00ae Core\u2122 i7-1065G7",
      "Processor family": "10th Gen Intel Core i7",
      "Processor Frequency": "Base frequency 1.30 GHz, Max frequency up to 3.90GHz",
      "Processor Cache": "8M",
      "Processor cores": "4 cores",
      "Ram": "8GB",
      "Ram Type": "DDR4-SDRAM",
      "Memory": "256GB SSD",
      "Memory slots": "2",
      "SSD Details": "PCIEG3X2 NVME",
      "Card Reader Integrated": "Yes",
      "Screen size": "15.6&quot;",
      "Panel Type": "IPS",
      "Aspect Ratio": "16:9",
      "HD Type": "Full HD",
      "Touchscreen": "No",
      "Display Resolution": "1920 x 1080 pixels",
      "Graphic Card": "Intel\u00ae Iris\u00ae Plus",
      "Graphics Memory": "Shared",
      "On-board Graphics Adapter": "Yes",
      "Graphics Memory Type": "Integrated",
      "Discrete Graphics Adapter": "Yes",
      "DVD": "NO DVD",
      "Wi-Fi Model": "802.11ac",
      "Ethernet LAN": "No",
      "Bluetooth": "Yes",
      "Built-in Microphone": "Yes",
      "Built-in Speakers": "2",
      "Front Camera": "Yes",
      "Operating system installed": "No",
      "Numeric Keypad": "Yes",
      "Pointing Device": "Touchpad",
      "Battery Technology": "Li-Polymer",
      "Battery Capacity": "52.5Wh",
        "Number of battery cells": "3",
        "Weight": "3.96 lbs (1.79 kg)",
        "Height": "17.9mm",
        "AC Adapter Power": "45W Round Tip",
        "Fingerprint Reader": "No"
    }
```

File name: raw.txt

Purpose: a file that contains th html source code of the page shown in Figure 4. (It is a huge text file so only some lines are shown below)

```
<div class="border-t mt-3"><a class="flex border-b py-2" href="https://shop.shpresa.al/product/lenovo-nb-ideapad-5-15-6-8gb-512gb-ssd-amd/"><img src="https://assets.shpresa.al/shop/2020/12/917fb83c-itd1216-300x300.jpg" class="w-20 h-20 object-cover" alt="Lenovo NB Ideapad 5 15.6&quot;, 8GB, 512GB SSD, AMD"><article class="pl-3 flex-1"><span class="text-lg font-bold block text-gray-800">Lenovo NB Ideapad 5 15.6", 8GB, 512GB SSD, AMD</span><span class="font-medium block text-gray-800 pb-2">ITD1378</span><div class="pb-2 flex"><span class="inline-block border rounded leading-normal uppercase text-xs font-bold px-1 bg-teal-100 border-teal-700 text-teal-700"><span>Në gjendje</span></span><div style="width: 4px;"></div></div><div><span class="price font-bold text-base"><ins class="block no-underline text-gray-800"><span><span>86,800</span>
<span>L</span></span></ins></span></div></article></a><a class="flex border-b py-2" href="https://shop.shpresa.al/product/dell-inspiron-3501-15-6-8gb-256gb-ssd-i5/"><img src="https://assets.shpresa.al/shop/2021/05/aae5b084-itd1345-1-300x300.jpg" class="w-20 h-20 object-cover" alt="Dell Inspiron 3501 15.6&quot;, 8GB, 256GB SSD, i5"><article class="pl-3 flex-1"><span class="text-lg font-bold block text-gray-800">Dell Inspiron 3501 15.6", 8GB, 256GB SSD, i5</span><span class="font-medium block text-gray-800 pb-2">ITD1374</span><div class="pb-2 flex"><span class="inline-block border rounded leading-normal uppercase text-xs font-bold px-1 bg-teal-100 border-teal-700 text-teal-700"><span>Në gjendje</span></span><div style="width: 4px;"></div></div><div><span class="price font-bold text-base"><ins class="block no-underline text-gray-800"><span><span>77,700</span>
```

File name: saved.bin

Purpose: this file saves a dictionary object for later use. The dictionary has all the original entried of data.json. file is not shown here as it is a binary file.

File name: DATASET_FINAL.xlsx

Purpose: an Excel file containing 5 sheets that represent the dataset in different stages. The contents are not shown here due to size.

File name: DATASET_FINAL.csv

Putpose: this file contains the same dataset as the final sheet on DATASET_FINAL.xlsx, but it saves it with comma separated values. This file is very important to the files that will be listed below.

```
no,brand,price,cpu_man,cpu_freq,cpu_type,n_cores,cache_size,ram_size,ssd_size,
display_tech,px_res,nits,width,g_card,bat_cap,bat_dur,ac_a_power,os,fingerprin
t
1,0.3571,983.18,1,0.784313725,0.881,0.3,0.6,0.4,8,8,0.613611111,0.6972,0.78034
6821,0.095547,0.500500501,0.71875,0.282608696,0.875,0
2,0.005,347.68,0.43,0.470588235,0.835,0.1,0.1,0.2,7,4,0.189722222,0.4,0.809248
555,0.095547,0.45045045,0.4375,0.282608696,0.688,0
3,0.005,438.7,0.43,0.666666667,0.832,0.1,0.2,0.2,7,4,0.375,0.44,0.901734104,0.
095547,0.45045045,0.45625,0.282608696,0.688,0
4,0.005,983.18,0.43,0.823529412,0.864,0.2,0.3,0.4,9,6,0.375,0.5,0.809248555,0.
095547,0.570570571,0.75,0.282608696,0.875,1
5,0.0128,794.58,0.43,0.823529412,0.793,0.2,0.4,0.4,8,6,0.375,0.822,0.768786127
,0.099153,0.510510511,0.6875,0.282608696,0.875,1
6,0.005,786.38,1,0.803921569,0.68,0.4,0.4,0.8,9,6,0.375,0.5,0.809248555,0.0955
47,0.525525526,1,0.282608696,0.875,1
7,0.005,819.18,0.43,0.823529412,0.864,0.2,0.3,0.4,8,6,0.375,0.6,0.768786127,0.
091941,0.48048048,0.65,0.282608696,0.875,1
8,0.0128,761.78,1,0.803921569,0.68,0.4,0.4,0.4,9,6,0.375,0.6,0.901734104,0.095
547,0.41041041,0.375,0.195652174,0.875,0
9,0.0128,786.38,1,0.803921569,0.68,0.4,0.4,0.8,8,6,0.375,0.44,0.901734104,0.09
5547,0.41041041,0.375,0.195652174,0.875,0
10,0.005,728.16,0.43,0.823529412,0.864,0.2,0.3,0.4,8,6,0.375,0.5,0.809248555,0
.095547,0.45045045,0.875,0.282608696,0.875,1
11,0.005,819.18,0.43,0.960784314,0.805,0.2,0.4,0.4,8,6,0.375,0.5,0.809248555,0
.095547,0.45045045,0.65625,0.282608696,0.875,1
12,0.0181,1311.18,1,0.862745098,0.919,0.4,0.4,0.8,10,6,0.375,0.6,0.901734104,0
.14963,0.510510511,0.453125,0.282608696,0.875,0
13,0.0128,1184.08,0.43,0.980392157,0.655,0.3,0.6,0.4,10,6,0.375,0.6,0.90173410
4,0.144222,0.690690691,0.616875,0.869565217,0,0
```

```
14,0.005,471.5,1,0.725490196,0.805,0.2,0.2,0.2,7,6,0.375,0.5,0.809248555,0.095547,0.525525526,1,0.282608696,0.875,1
15,1,1065.18,0.12,0.62745098,0.899,0.4,0.6,0.4,8,7,0.740740741,0.8,0.751445087,0.281004,0.499499499,0.9375,0.130434783,1,0
16,0.0181,883.14,0.43,0.882352941,0.864,0.2,0.6,0.4,9,6,0.375,0.5,0.901734104,0.353619,0.510510511,0.510625,0.565217391,0,1
17,0.005,711.76,0.43,0.823529412,0.864,0.2,0.3,0.8,9,6,0.740740741,0.6,0.768786127,0.095547,0.560560561,0.6875,0.282608696,0.875,1
18,0.0128,1147.18,0.43,0.980392157,0.655,0.3,0.6,0.4,9,6,0.375,0.6,0.901734104,0.144222,0.690690691,0.3125,0.869565217,0,0
19,0.005,736.36,1,0.784313725,0.881,0.3,0.4,0.4,8,4,0.375,0.5,0.809248555,0.095547,0.45045045,0.53125,0.282608696,0.875,1
20,0.005,327.18,0.43,0.607843137,0.764,0.2,0.2,0.2,7,4,0.189722222,0.478,0.809248555,0.091941,0.32032032,0.5625,0.195652174,0.688,0
21,0.0005,628.94,1,0.725490196,0.881,0.2,0.2,0.4,9,6,0.375,0.536,0.901734104,0.099153,0.32032032,0.34375,0.282608696,0,1
22,0.0181,637.14,0.43,0.823529412,0.793,0.2,0.4,0.4,8,4,0.375,0.44,0.901734104,0.099153,0.42042042,0.53125,0.282608696,0.875,0
23,0.005,573.18,1,0.725490196,0.881,0.2,0.2,0.4,8,4,0.375,0.44,0.901734104,0.106364,0.35035035,0.3125,0.282608696,0.875,0
24,0.005,573.18,1,0.725490196,0.805,0.2,0.1,0.4,8,6,0.375,0.44,0.901734104,0.106364,0.35035035,0.3125,0.282608696,0.875,0
25,0.005,865.1,0.43,0.764705882,0.655,0.2,0.4,0.4,9,6,0.375,0.5,0.901734104,0.11351,0.525525526,0.609375,0.282608696,0.875,1
26,0.005,983.18,1,0.823529412,0.68,0.4,0.4,0.4,9,6,0.375,0.5,0.901734104,0.353619,0.45045045,0.25,0.586956522,0.875,0
27,0.0019,1548.16,0.43,0.980392157,0.655,0.3,0.6,0.8,10,6,0.375,0.524,0.901734104,0.178475,0.510510511,0.28125,0.782608696,0.875,0
28,0.0181,1393.18,0.43,0.980392157,0.655,0.3,0.6,0.8,10,6,0.375,0.6,0.901734104,0.178475,0.860860861,0.21875,0.586956522,0.875,0
29,0.0005,1598.18,0.43,0.980392157,0.655,0.3,0.6,0.8,9,6,0.375,0.5,1,0.120771,0.660660661,0.421875,1,0,0
30,0.0128,529.72,0.43,0.803921569,1,0.1,0.3,0.4,8,2,0.189722222,0.398,0.901734104,0.095547,0.41041041,0.71875,0.195652174,0.875,0
31,0.005,711.76,1,0.803921569,0.68,0.4,0.4,0.4,9,3,0.375,0.6,0.901734104,0.095547,0.45045045,0.7125,0.282608696,0,1
32,0.0181,637.14,0.43,0.705882353,0.864,0.2,0.3,0.4,8,4,0.375,0.44,0.901734104,0.095547,0.42042042,0.5125,0.282608696,0.875,0
33,0.005,327.18,1,0.647058824,0.813,0.1,0.2,0.2,7,4,0.375,0.454,0.809248555,0.095547,0.3003003,0.53125,0.282608696,0.875,0
34,0.005,737.18,0.43,0.764705882,0.655,0.2,0.4,0.6,9,4,0.375,0.44,0.901734104,0.178475,0.35035035,0.7125,0.282608696,0,0
35,0.005,487.9,0.43,0.803921569,0.832,0.1,0.2,0.4,8,4,0.375,0.436,0.901734104,0.095547,0.35035035,0.46875,0.282608696,0,0
36,0.0005,1393.18,0.43,0.941176471,0.805,0.2,0.6,0.8,9,1,0.375,0.53,0.901734104,0.178475,0.760760761,0.625,0.434782609,0.875,0
```

```
37,0.005,653.54,1,0.784313725,0.881,0.3,0.4,0.4,9,3,0.375,0.5,0.901734104,0.09
5547,0.45045045,0.7125,0.282608696,0,1
38,0.005,703.56,1,0.803921569,0.68,0.4,0.4,0.4,8,4,0.375,0.5,0.901734104,0.095
547,0.35035035,0.6,0.282608696,0,0
39,0.005,1275.1,1,0.823529412,0.68,0.4,0.4,0.8,9,6,0.375,0.6,0.901734104,0.153
257,0.600600601,0.453125,1,0.875,0
40,0.0005,1851.8,0.43,0.941176471,0.805,0.2,0.6,0.8,10,1,0.375,0.53,0.90173410
4,0.178482,0.760760761,0.625,0.434782609,0.875,0
41,0.0005,1475.18,1,0.843137255,0.919,0.4,0.4,0.8,10,1,0.375,0.646,0.809248555
,0.153257,0.760760761,0.6875,0.782608696,0.875,0
42,0.0128,521.52,0.43,0.803921569,1,0.1,0.3,0.4,8,4,0.189722222,0.398,0.809248
555,0.095547,0.41041041,0.390625,0.195652174,0.875,0
43,0.0128,637.14,0.43,0.803921569,1,0.1,0.3,0.4,8,4,0.189722222,0.42,0.8092485
55,0.095547,0.41041041,0.46875,0.195652174,0.875,0
44,0.005,695.36,1,0.784313725,0.68,0.2,0.2,0.6,9,6,0.375,0.444,0.901734104,0.0
99178,0.525525526,0.5,0.195652174,0,0
45,0.3571,1147.18,0.08,0.588235294,0.788,0.4,0.1,0.4,8,8,1,0.9,0.751445087,0.0
2551,0.500500501,0.8125,0.282608696,0.875,0
46,0.005,736.36,1,0.784313725,0.881,0.3,0.4,0.4,8,4,0.375,0.464,0.809248555,0.
095547,0.45045045,0.53125,0.282608696,0.875,1
47,0.0181,1393.18,0.43,0.980392157,0.655,0.3,0.6,0.8,10,6,0.375,0.6,0.90173410
4,0.153257,0.860860861,0.296875,0.586956522,0.875,0
48,0.0019,901.18,0.43,0.882352941,0.864,0.2,0.4,0.4,7,6,0.375,0.506,1,0.353619
,0.510510511,0.28125,0.652173913,0.875,0
49,0.0019,883.14,0.43,0.882352941,0.864,0.2,0.4,0.4,9,6,0.375,0.506,1,0.353619
,0.510510511,0.28125,0.652173913,0.875,0
50,0.0181,653.54,0.43,0.823529412,0.793,0.2,0.4,0.4,8,6,0.375,0.466,0.90173410
4,0.099153,0.530530531,0.5625,0.282608696,0.875,1
51,0.0005,307.5,0.43,0.549019608,0.712,0.1,0.2,0.2,7,6,0.189722222,0.442,0.809
248555,0.95547,0.42042042,0.6875,0.143478261,0,0
52,0.0019,1275.1,0.43,0.921568627,0.655,0.3,0.6,0.8,9,6,0.375,0.564,0.80924855
5,0.353619,0.520520521,0.8375,0.391304348,0.875,1
53,0.0019,1307.9,0.43,0.921568627,0.655,0.3,0.6,0.8,9,6,0.375,0.564,0.80924855
5,0.353619,0.520520521,0.8375,0.391304348,0.875,1
54,0.0019,1184.08,0.43,0.980392157,0.655,0.3,0.6,0.8,9,6,0.375,0.654,0.9017341
04,0.353619,0.510510511,0.198125,0.782608696,0.875,1
55,0.0128,562.52,1,0.784313725,0.881,0.3,0.4,0.4,8,6,0.375,0.398,0.809248555,0
.095547,0.41041041,0.3875,0.195652174,0.875,0
56,0.005,736.36,0.43,0.764705882,0.655,0.2,0.4,0.4,8,3,0.375,0.6,0.901734104,0
.11351,0.45045045,0.7125,0.282608696,0,1
57,0.005,471.5,0.43,0.666666667,0.832,0.1,0.2,0.4,7,4,0.375,0.44,0.901734104,0
.095547,0.35035035,0.7125,0.282608696,0,0
58,0.0005,596.14,0.43,0.705882353,0.864,0.2,0.3,0.4,8,4,0.375,0.41,0.901734104
,0.095547,0.37037037,0.33125,0.195652174,0,1
59,0.0019,1335.78,0.43,1,0.655,0.4,0.8,0.8,9,6,0.375,0.47,0.901734104,0.153257
,1,0.625,0.782608696,0.875,1
```

```
60,0.005,532.18,1,0.725490196,0.881,0.2,0.2,0.4,8,4,0.375,0.436,0.901734104,0.
106364,0.35035035,0.3125,0.282608696,0,0
61,1,1521.1,0.12,0.62745098,0.899,0.4,0.6,0.4,9,7,0.740740741,1,0.751445087,1,
0.582582583,0.83125,0.282608696,1,1
62,0.0128,819.18,1,0.803921569,0.68,0.4,0.2,0.8,9,5,0.375,0.46,0.901734104,0.0
95547,0.41041041,0.5625,0.282608696,0.875,0
63,0.005,537.92,1,0.68627451,0.805,0.1,0.2,0.4,9,4,0.375,0.436,0.901734104,0.0
95547,0.45045045,0.3125,0.282608696,0,0
64,0.005,794.58,0.43,0.764705882,0.655,0.2,0.4,0.6,9,6,0.375,0.6,0.901734104,0
.11351,0.45045045,0.7125,0.282608696,0.875,1
65,0.005,570.72,0.43,0.705882353,0.864,0.2,0.3,0.4,8,4,0.375,0.436,0.901734104
,0.095547,0.35035035,0.3125,0.282608696,0,0
66,0.005,1193.1,0.43,0.980392157,0.655,0.3,0.6,0.8,9,6,0.375,0.6,0.901734104,0
.144222,0.800800801,0.4375,0.739130435,0.875,0
67,0.005,1311.18,0.43,0.980392157,0.655,0.3,0.6,0.8,9,6,0.375,0.574,0.90173410
4,0.153257,0.800800801,0.4375,0.739130435,0.875,0
68,0.0128,778.18,0.43,0.882352941,0.864,0.2,0.4,0.4,8,6,0.375,0.5,0.901734104,
0.353619,0.525525526,0.4,0.565217391,0.875,0
69,0.005,983.18,0.43,0.921568627,0.805,0.2,0.6,0.6,9,6,0.375,0.532,0.809248555
,0.099153,0.510510511,0.78125,0.282608696,0.875,1
70,0.0005,430.5,0.43,0.666666667,0.832,0.1,0.2,0.2,7,6,0.189722222,0.514,0.901
734104,0.095547,0.37037037,0.340625,0.282608696,0,1
```

File name: view_heatmap.py

Purpose: a file put together in order to interpret the data shown in DATASET_FINAL.csv file. It shows the heatmap of the correlation between features (Figure 12).

```python
# script to view the correlation matric of our data shown as a heatmap plot
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # plotting
import seaborn as sns # contains the heatmap function that returns a subplot
# read the data from csv file and save it in a dataframe
df = pd.read_csv('DATASET_FINAL.csv')
# remove the column named no as it is just for indexing
df.pop('no')
# print 5 first entries for the dataframe
print(df.head())
# define a correlation matrix for the data using the Pearson's method
corr = df.corr(method='pearson')
# print the 5 first entries of the correlation matrix
corr.head()
# create a heatmap subplot
ax = sns.heatmap(corr, linewidths=1)
# show the subplot
plt.show()
```

File name: graph_with_2.py

Purpose: show and save an image in images2 folder according to input from user. The image contains 2 subplots: the one on the left is a subplot of a feature against price (raw data), the one in the right is the same as the graph on the left but FCM with c = 2 is applied on the data. (Data is separated between 2 clusters with centroids shown as red circles)

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning) # ignore warnin
gs
from fcmeans import FCM # Fuzzy C-Means Algorithm implementation in Python
from seaborn import scatterplot as scatter # building a scatterplot of data
from matplotlib import pyplot as plt # plotting the scatterplot
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np # data processing and normalization (e.g. arrays, np.linalg
.norm)
# ask the user for input and handle miss input
feature = input("What feature are you comparing to price?\nbrand, price, cpu_m
an, cpu_freq, cpu_type, n_cores, cache_size, ram_size, ssd_size, display_tech,
 px_res, nits, width, g_card, bat_cap, bat_dur, ac_a_power, os, fingerprint\n"
)
lst = ["brand", "price", "cpu_man", "cpu_freq", "cpu_type", "n_cores", "cache_
size", "ram_size", "ssd_size", "display_tech", "px_res", "nits", "width", "g_c
ard", "bat_cap", "bat_dur", "ac_a_power", "os", "fingerprint"]
if feature not in lst:
    print("Invalid input! Make sure that your input is one of: brand, price, c
pu_man, cpu_freq, cpu_type, n_cores, cache_size, ram_size, ssd_size, display_t
ech, px_res, nits, width, g_card, bat_cap, bat_dur, ac_a_power, os, fingerprin
t")
    exit(1)
# load the data
df = pd.read_csv('DATASET_FINAL.csv')
# create an empty array that will hold the values for 2 columns, price and ano
ther feature. This array liiks like: [[price1, price2, ..., price70],[other_fe
ature1, other_feature2, ..., other_feature70]]
a = np.empty([2,70])
# normalize the price column
norm = np.max(df['price'])
normal_price = df['price']/norm
# fill the values of the empty array previously created
a[0] = normal_price
a[1] = df[feature]
# conditions to normalize unnormalized data
if feature == 'ssd_size':
    a[1] = df[feature]/10
elif feature == 'display_tech':
    a[1] = df[feature]/8
elif feature == 'price':
    a[1] = normal_price
else:
    pass
# transpose a to create an array with 70 rows and 2 columns where first column
 has the price data and the second coumn has data from another feature; array
composition: [[normal_price1, other_feature1], [normal_price2, other_feature2]
, ..., [normal_price70, other_feature70]]
X = np.transpose(a)
# fit the fuzzy-c-means
fcm = FCM(n_clusters=2)
fcm.fit(X)
# outputs
fcm_centers = fcm.centers
fcm_labels  = fcm.u.argmax(axis=1)
# plot result
f, axes = plt.subplots(1, 2)
f.tight_layout(pad = 3)
plt.setp(axes, xlim=(-0.15, 1.15), ylim=(-0.15, 1.15))
for ax in axes:
    ax.set(adjustable='box', aspect='equal')
scatter(X[:,0], X[:,1], ax=axes[0])
scatter(X[:,0], X[:,1], ax=axes[1], hue=fcm_labels)
scatter(fcm_centers[:,0], fcm_centers[:,1], ax=axes[1], marker='s', color='r',
 s=50)
f.suptitle(feature, size=16)
for ax in axes.flat:
    ax.set(xlabel="price", ylabel=feature)
# save the graph as a jpg image in images2 folder, indicating we used 2 cluste
rs
save_path = 'images2/'+feature+'_price.jpg'
plt.savefig(save_path, format='jpg')
plt.show()
```

File name: graph_with_3.py

Purpose: show and save an image in image3 folder according to input from user. The image contains 2 subplots: the one on the left is a subplot of a feature against price (raw data), the one in the right is the same as the graph on the left, but FCM with c = 3 is applied on the data. (Data is separated between 3 clusters with centroids shown as black crosses)

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning) # ignore warnin
gs
from fcmeans import FCM # Fuzzy C-Means Algorithm implementation in Python
from seaborn import scatterplot as scatter # building a scatterplot of data
from matplotlib import pyplot as plt # plotting the scatterplot
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np # data processing and normalization (e.g. arrays, np.linalg
.norm)
# ask the user for input and handle miss input
feature = input("What feature are you comparing to price?\nbrand, price, cpu_m
an, cpu_freq, cpu_type, n_cores, cache_size, ram_size, ssd_size, display_tech,
 px_res, nits, width, g_card, bat_cap, bat_dur, ac_a_power, os, fingerprint\n"
)
lst = ["brand", "price", "cpu_man", "cpu_freq", "cpu_type", "n_cores", "cache_
size", "ram_size", "ssd_size", "display_tech", "px_res", "nits", "width", "g_c
ard", "bat_cap", "bat_dur", "ac_a_power", "os", "fingerprint"]
if feature not in lst:
    print("Invalid input! Make sure that your input is one of: brand, price, c
pu_man, cpu_freq, cpu_type, n_cores, cache_size, ram_size, ssd_size, display_t
ech, px_res, nits, width, g_card, bat_cap, bat_dur, ac_a_power, os, fingerprin
t")
    exit(1)
# Load the data
df = pd.read_csv('DATASET_FINAL.csv')
# create an empty array that will hold the values for 2 columns, price and ano
ther feature. This array liiks like: [[price1, price2, ..., price70],[other_fe
ature1, other_feature2, ..., other_feature70]]
a = np.empty([2,70])
# normalize the price column
norm = np.max(df['price'])
normal_price = df['price']/norm
# fill the values of the empty array previously created
a[0] = normal_price
a[1] = df[feature]
# conditions to normalize unnormalized data
if feature == 'ssd_size':
    a[1] = df[feature]/10
elif feature == 'display_tech':
    a[1] = df[feature]/8
elif feature == 'price':
    a[1] = normal_price
else:
    pass
# transpose a to create an array with 70 rows and 2 columns where first column
 has the price data and the second coumn has data from another feature; array
composition: [[normal_price1, other_feature1], [normal_price2, other_feature2]
, ..., [normal_price70, other_feature70]]
X = np.transpose(a)
# fit the fuzzy-c-means
fcm = FCM(n_clusters=3)
fcm.fit(X)
# outputs
fcm_centers = fcm.centers
fcm_labels  = fcm.u.argmax(axis=1)
# plot result
f, axes = plt.subplots(1, 2)
f.tight_layout(pad = 3)
plt.setp(axes, xlim=(-0.15, 1.15), ylim=(-0.15, 1.15))
for ax in axes:
    ax.set(adjustable='box', aspect='equal')
scatter(X[:,0], X[:,1], ax=axes[0])
scatter(X[:,0], X[:,1], ax=axes[1], hue=fcm_labels, palette="prism")
scatter(fcm_centers[:,0], fcm_centers[:,1], ax=axes[1], marker='X', color='bla
ck', s=50)
f.suptitle(feature, size=16)
for ax in axes.flat:
    ax.set(xlabel="price", ylabel=feature)
# save the graph as a jpg image in images2 folder, indicating we used 2 cluste
rs
save_path = 'images3/'+feature+'_price3.jpg'
plt.savefig(save_path, format='jpg')
plt.show()
```

File name: automated2.py

Purpose: a file to populate images2 folder with 19 pictures described previously in graph_with_2.py file’s purpose

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning) # ignore warnin
gs
from fcmeans import FCM # Fuzzy C-Means Algorithm implementation in Python
from seaborn import scatterplot as scatter # building a scatterplot of data
from matplotlib import pyplot as plt # plotting the scatterplot
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np # data processing and normalization (e.g. arrays, np.linalg
.norm)
lst = ["brand", "price", "cpu_man", "cpu_freq", "cpu_type", "n_cores", "cache_
size", "ram_size", "ssd_size", "display_tech", "px_res", "nits", "width", "g_c
ard", "bat_cap", "bat_dur", "ac_a_power", "os", "fingerprint"]
# load the data
df = pd.read_csv('DATASET_FINAL.csv')
# create an empty array that will hold the values for 2 columns, price and ano
ther feature. This array liiks like: [[price1, price2, ..., price70],[other_fe
ature1, other_feature2, ..., other_feature70]]
a = np.empty([2,70])
# normalize the price column
norm = np.max(df['price'])
normal_price = df['price']/norm
# fill the values of the empty array previously created
# a[0] = normal_price
a[0] = normal_price
for feature in lst:
    a[1] = df[feature]
    if feature == 'ssd_size':
        a[1] = df[feature]/10
    elif feature == 'display_tech':
        a[1] = df[feature]/8
    elif feature == 'price':
        a[1] = normal_price
    else:
        pass
    # transpose a to create an array with 70 rows and 2 columns where first co
lumn has the price data and the second coumn has data from another feature; ar
ray composition: [[normal_price1, other_feature1], [normal_price2, other_featu
re2], ..., [normal_price70, other_feature70]]
    X = np.transpose(a)
    # fit the fuzzy-c-means
    fcm = FCM(n_clusters=2)
    fcm.fit(X)
    # outputs
    fcm_centers = fcm.centers
    fcm_labels  = fcm.u.argmax(axis=1)
    # plot result
    f, axes = plt.subplots(1, 2)
    f.tight_layout(pad = 3)
    plt.setp(axes, xlim=(-0.15, 1.15), ylim=(-0.15, 1.15))
    for ax in axes:
        ax.set(adjustable='box', aspect='equal')
    scatter(X[:,0], X[:,1], ax=axes[0])
    scatter(X[:,0], X[:,1], ax=axes[1], hue=fcm_labels)
    scatter(fcm_centers[:,0], fcm_centers[:,1], ax=axes[1], marker='s', color=
'r', s=50)
    f.suptitle(feature, size=16)
    for ax in axes.flat:
        ax.set(xlabel="price", ylabel=feature)
    # save the graph as a jpg image in images2 folder, indicating we used 2 cl
usters
    save_path = 'images2/'+feature+'_price.jpg'
    plt.savefig(save_path, format='jpg')
    # plt.show()
```

File name: automated3.py

Purpose: a file to populate images3 folder with 19 pictures described previously in graph_with_3.py file's purpose

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning) # ignore warnin
gs
from fcmeans import FCM # Fuzzy C-Means Algorithm implementation in Python
from seaborn import scatterplot as scatter # building a scatterplot of data
from matplotlib import pyplot as plt # plotting the scatterplot
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np # data processing and normalization (e.g. arrays, np.linalg
.norm)
lst = ["brand", "price", "cpu_man", "cpu_freq", "cpu_type", "n_cores", "cache_
size", "ram_size", "ssd_size", "display_tech", "px_res", "nits", "width", "g_c
ard", "bat_cap", "bat_dur", "ac_a_power", "os", "fingerprint"]
# load the data
df = pd.read_csv('DATASET_FINAL.csv')
# create an empty array that will hold the values for 2 columns, price and ano
ther feature. This array liiks like: [[price1, price2, ..., price70],[other_fe
ature1, other_feature2, ..., other_feature70]]
```

```
a = np.empty([2,70])
# normalize the price column
norm = np.max(df['price'])
normal_price = df['price']/norm
# fill the values of the empty array previously created
# a[0] = normal_price
a[0] = normal_price
for feature in lst:
    a[1] = df[feature]
    if feature == 'ssd_size':
        a[1] = df[feature]/10
    elif feature == 'display_tech':
        a[1] = df[feature]/8
    elif feature == 'price':
        a[1] = normal_price
    else:
        pass
    # transpose a to create an array with 70 rows and 2 columns where first co
lumn has the price data and the second coumn has data from another feature; ar
ray composition: [[normal_price1, other_feature1], [normal_price2, other_featu
re2], ..., [normal_price70, other_feature70]]
    X = np.transpose(a)
    # fit the fuzzy-c-means
    fcm = FCM(n_clusters=3)
    fcm.fit(X)
    # outputs
    fcm_centers = fcm.centers
    fcm_labels  = fcm.u.argmax(axis=1)
    # plot result
    f, axes = plt.subplots(1, 2)
    f.tight_layout(pad = 3)
    plt.setp(axes, xlim=(-0.15, 1.15), ylim=(-0.15, 1.15))
    for ax in axes:
        ax.set(adjustable='box', aspect='equal')
    scatter(X[:,0], X[:,1], ax=axes[0])
    scatter(X[:,0], X[:,1], ax=axes[1], hue=fcm_labels, palette="prism")
    scatter(fcm_centers[:,0], fcm_centers[:,1], ax=axes[1], marker='X', color=
'black', s=50)
    f.suptitle(feature, size=16)
    for ax in axes.flat:
        ax.set(xlabel="price", ylabel=feature)
    # save the graph as a jpg image in images2 folder, indicating we used 2 cl
usters
    save_path = 'images3/'+feature+'_price3.jpg'
    plt.savefig(save_path, format='jpg')
    # plt.show()
```

File name: automatedR2.py

Purpose: file to populate imagesR2 folder that contains graphs showing regression line between 2 cluster centroids and lines x = y, x = -y in order to make it easier to evaluate the effect that the feature has on price

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning) # ignore warnin
gs
from fcmeans import FCM # Fuzzy C-Means Algorithm implementation in Python
from seaborn import scatterplot as scatter # building a scatterplot of data
from matplotlib import pyplot as plt # plotting the scatterplot
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np # data processing and normalization (e.g. arrays, np.linalg
.norm)
lst = ["brand", "price", "cpu_man", "cpu_freq", "cpu_type", "n_cores", "cache_
size", "ram_size", "ssd_size", "display_tech", "px_res", "nits", "width", "g_c
ard", "bat_cap", "bat_dur", "ac_a_power", "os", "fingerprint"]
# load the data
df = pd.read_csv('DATASET_FINAL.csv')
# create an empty array that will hold the values for 2 columns, price and ano
ther feature. This array liiks like: [[price1, price2, ..., price70],[other_fe
ature1, other_feature2, ..., other_feature70]]
a = np.empty([2,70])
# normalize the price column
norm = np.max(df['price'])
normal_price = df['price']/norm
# fill the values of the empty array previously created
# a[0] = normal_price
a[0] = normal_price
for feature in lst:
    a[1] = df[feature]
    if feature == 'ssd_size':
        a[1] = df[feature]/10
    elif feature == 'display_tech':
        a[1] = df[feature]/8
    elif feature == 'price':
        a[1] = normal_price
    else:
        pass
    # transpose a to create an array with 70 rows and 2 columns where first co
lumn has the price data and the second coumn has data from another feature; ar
ray composition: [[normal_price1, other_feature1], [normal_price2, other_featu
re2], ..., [normal_price70, other_feature70]]
    X = np.transpose(a)
    # fit the fuzzy-c-means
    fcm = FCM(n_clusters=2)
    fcm.fit(X)
    # outputs
    fcm_centers = fcm.centers
    fcm_labels  = fcm.u.argmax(axis=1)
    # plot result
    f, axes = plt.subplots(1, 1)
    f.tight_layout(pad = 3)
    plt.setp(axes, xlim=(-0.15, 1.15), ylim=(-0.15, 1.15))
    axes.set(adjustable='box', aspect='equal')
    scatter(X[:,0], X[:,1], hue=fcm_labels)
    scatter(fcm_centers[:,0], fcm_centers[:,1], marker='s', color='r', s=50)
    m, b = np.polyfit(fcm_centers[:,0], fcm_centers[:,1], 1)
    x = fcm_centers[:,0]
    plt.plot(x, m*x + b, 'g')
    axes.plot([0,1],[0,1],  transform=axes.transAxes)
    axes.plot([0,1],[1,0],  transform=axes.transAxes)
    f.suptitle(feature, size=16)
    axes.set(xlabel="price", ylabel=feature)
    # save the graph as a jpg image in images2 folder, indicating we used 2 cl
usters
    save_path = 'imagesR2/'+feature+'_priceR2.jpg'
    plt.savefig(save_path, format='jpg')
    # plt.show()
```

File name: automatedR3.py

Purpose: file to populate imagesR3 folder that contains graphs showing regression line between 3 cluster centroids and lines x = y, x = -y in order to make it easier to evaluate the effect that the feature has on price

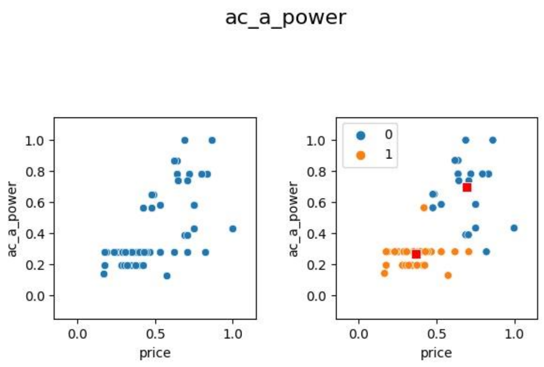
```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning) # ignore warnin
gs
from fcmeans import FCM # Fuzzy C-Means Algorithm implementation in Python
from seaborn import scatterplot as scatter # building a scatterplot of data
from matplotlib import pyplot as plt # plotting the scatterplot
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np # data processing and normalization (e.g. arrays, np.linalg
.norm)
lst = ["brand", "price", "cpu_man", "cpu_freq", "cpu_type", "n_cores", "cache_
size", "ram_size", "ssd_size", "display_tech", "px_res", "nits", "width", "g_c
ard", "bat_cap", "bat_dur", "ac_a_power", "os", "fingerprint"]
# load the data
df = pd.read_csv('DATASET_FINAL.csv')
```

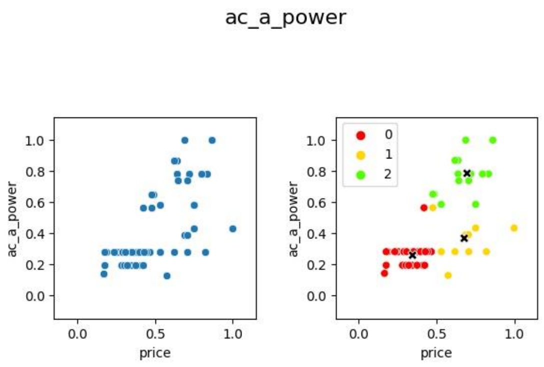
```
# create an empty array that will hold the values for 2 columns, price and ano
ther feature. This array liiks like: [[price1, price2, ..., price70],[other_fe
ature1, other_feature2, ..., other_feature70]]
a = np.empty([2,70])
# normalize the price column
norm = np.max(df['price'])
normal_price = df['price']/norm
# fill the values of the empty array previously created
# a[0] = normal_price
a[0] = normal_price
for feature in lst:
    a[1] = df[feature]
    if feature == 'ssd_size':
        a[1] = df[feature]/10
    elif feature == 'display_tech':
        a[1] = df[feature]/8
    elif feature == 'price':
        a[1] = normal_price
    else:
        pass
    X = np.transpose(a)
    # fit the fuzzy-c-means
    fcm = FCM(n_clusters=3)
    fcm.fit(X)
    # outputs
    fcm_centers = fcm.centers
    fcm_labels  = fcm.u.argmax(axis=1)
    # plot result
    f, axes = plt.subplots(1, 1)
    f.tight_layout(pad = 3)
    plt.setp(axes, xlim=(-0.15, 1.15), ylim=(-0.15, 1.15))
    axes.set(adjustable='box', aspect='equal')
    scatter(X[:,0], X[:,1], hue=fcm_labels, palette="prism")
    scatter(fcm_centers[:,0], fcm_centers[:,1], marker='X', color='black', s=5
0)
    m, b = np.polyfit(fcm_centers[:,0], fcm_centers[:,1], 1)
    x = fcm_centers[:,0]
    plt.plot(x, m*x + b, 'g')
    axes.plot([0,1],[0,1],  transform=axes.transAxes)
    axes.plot([0,1],[1,0],  transform=axes.transAxes)
    f.suptitle(feature, size=16)
    axes.set(xlabel="price", ylabel=feature)

    # save the graph as a jpg image in images2 folder, indicating we used 2 cl
usters
    save_path = 'imagesR3/'+feature+'_priceR3.jpg'
    plt.savefig(save_path, format='jpg')
```

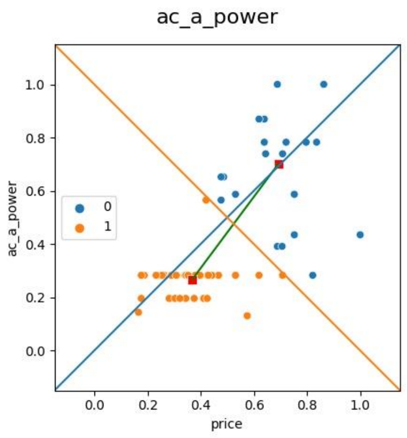
APPENDIX B

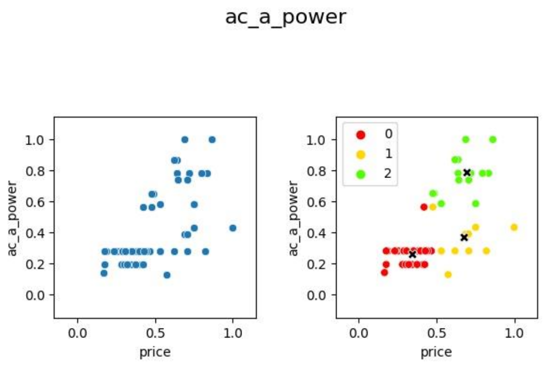
Clustering Outputs

AC adapter power effect on price

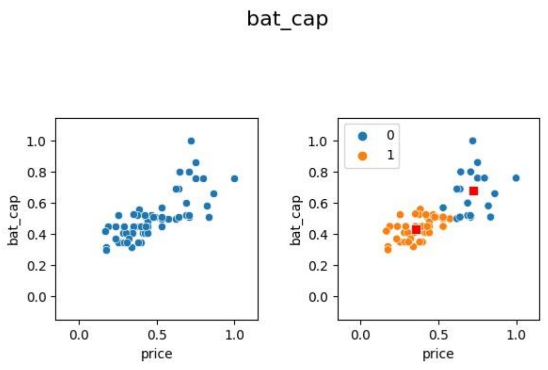


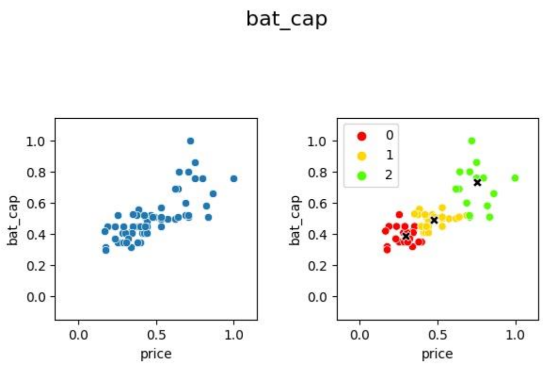


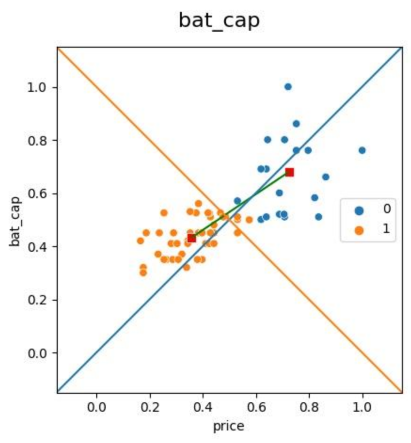


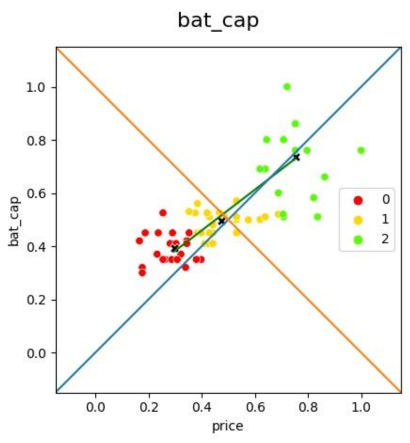


Battery capacity effect on power.

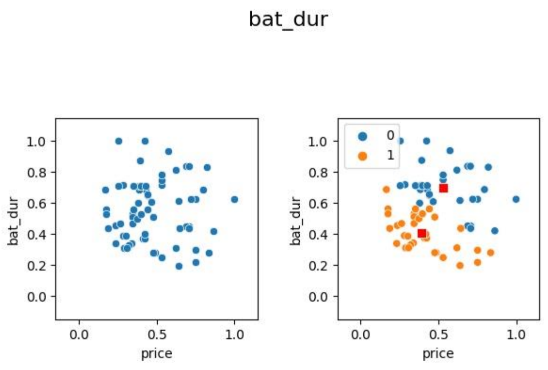


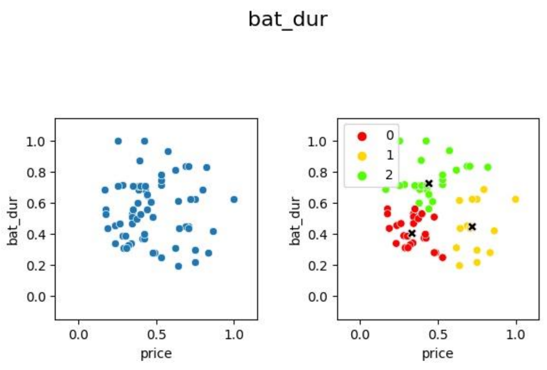


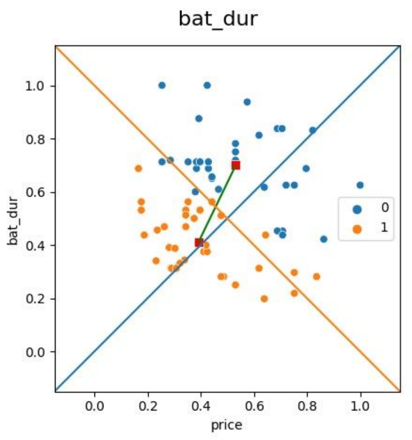


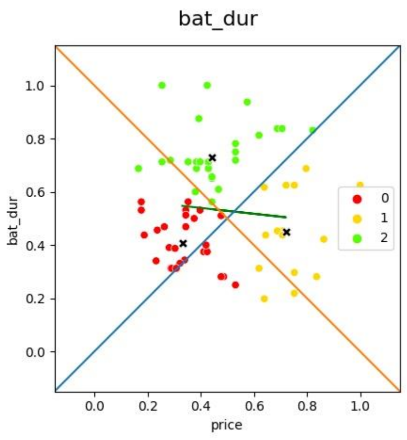


Battery durability effect on price.

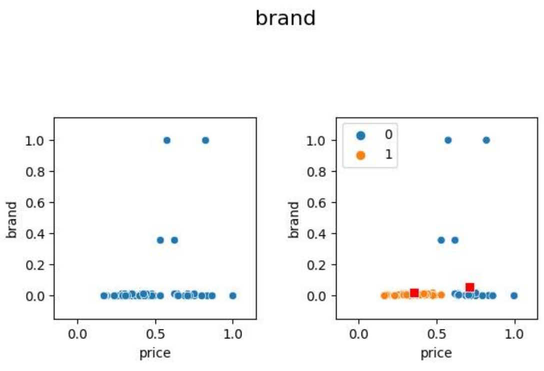


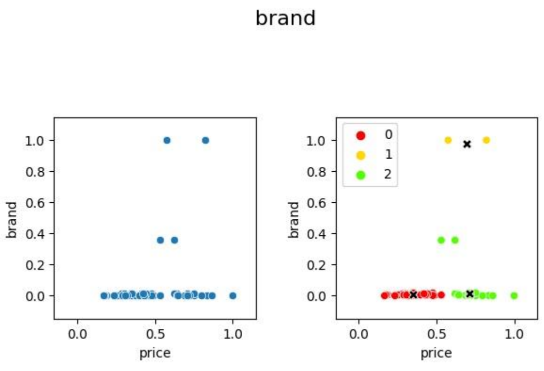


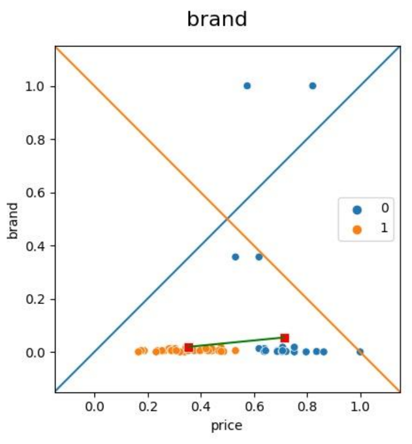


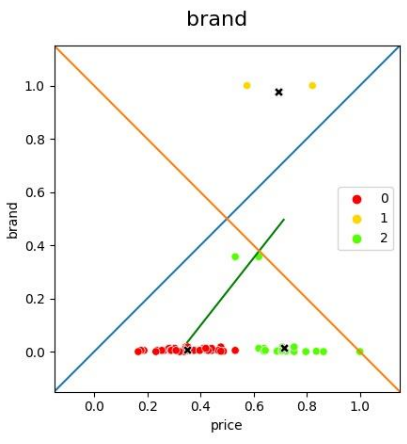


Laptop brand effect on price.

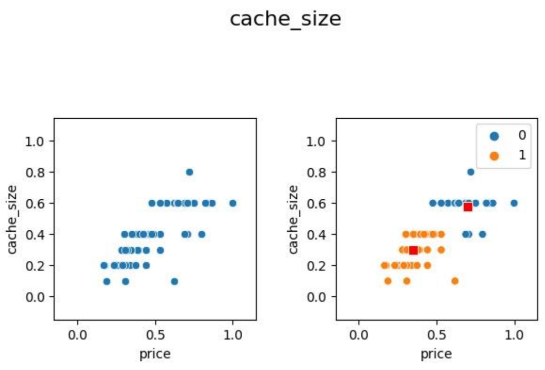


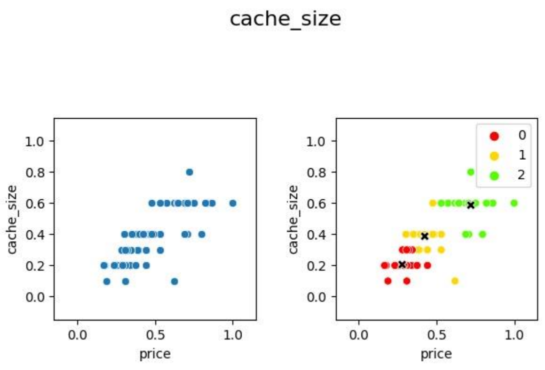


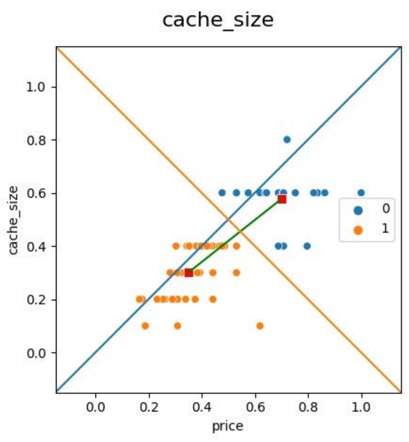


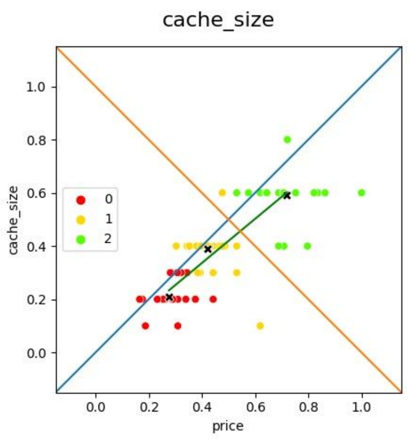


Cache size effect on price.

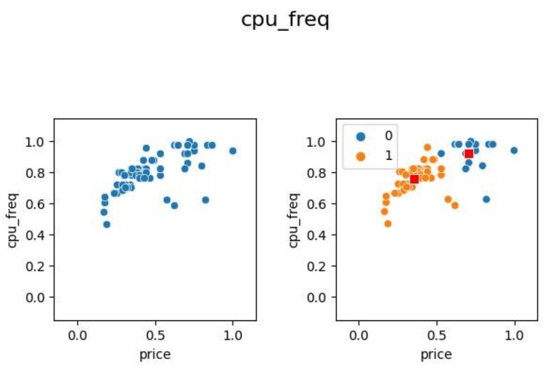


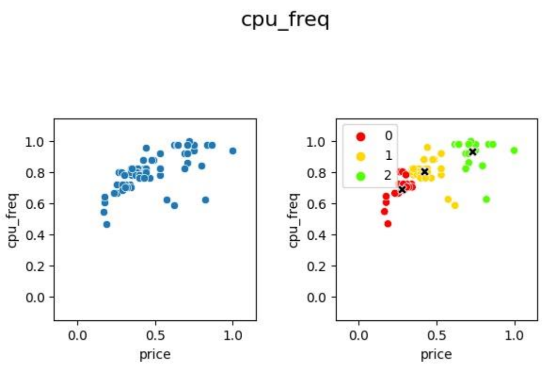


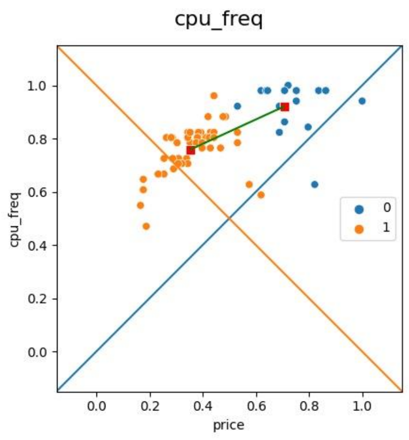


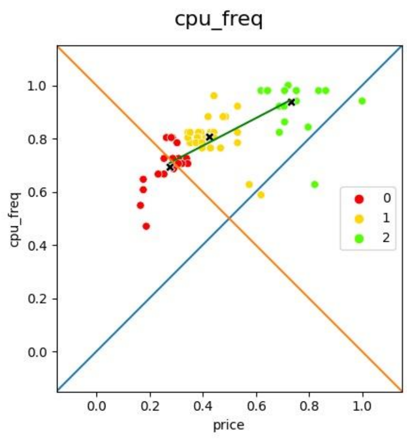


CPU frequency effect on price.

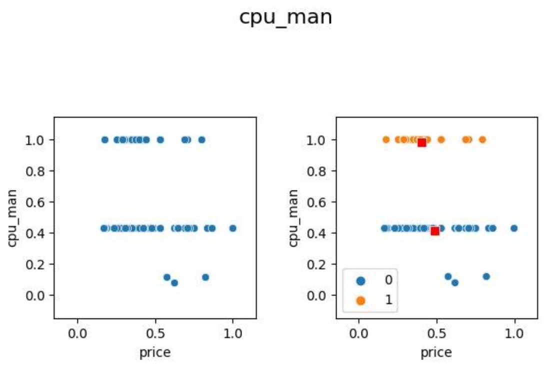


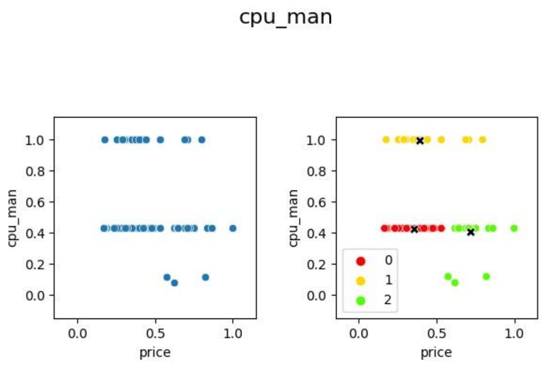


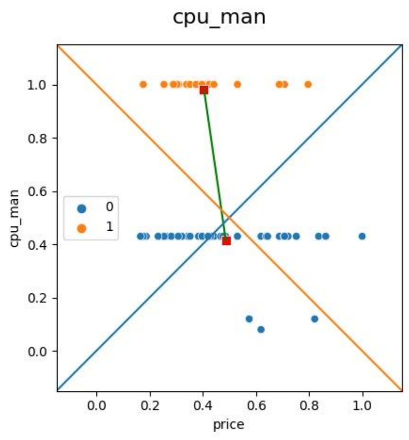


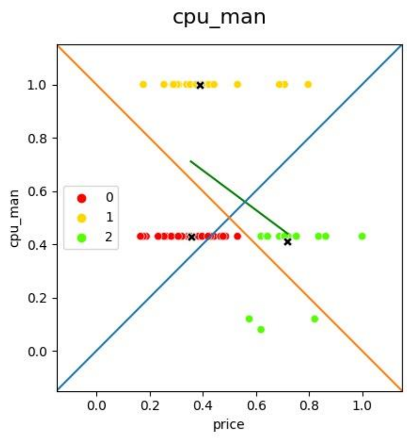


CPU manufacturer effect on price.

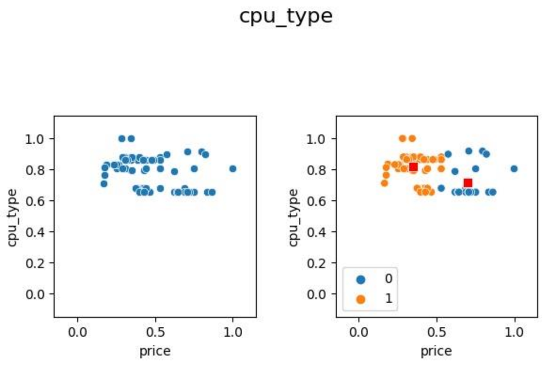


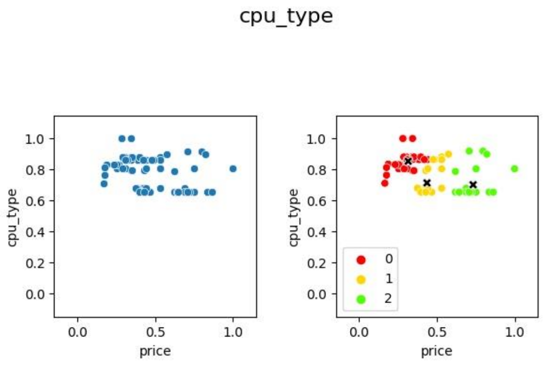


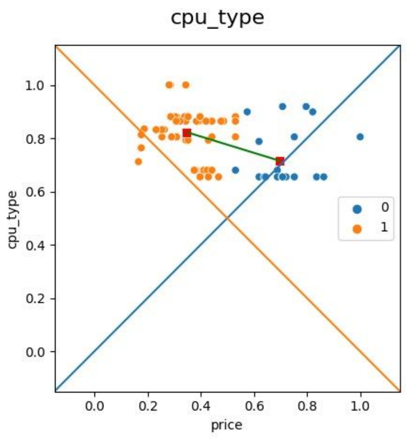


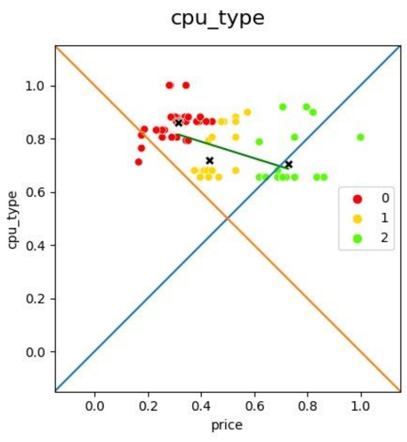


CPU type effect on price.

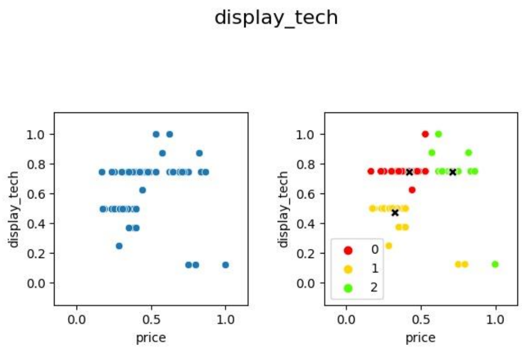


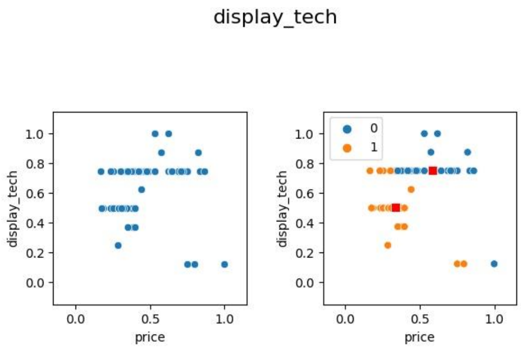


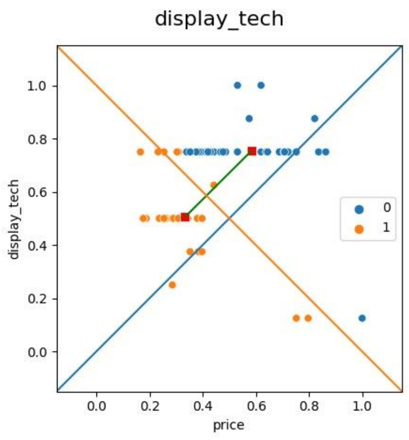


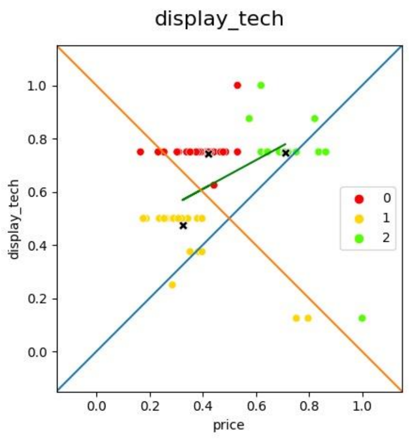


Display technology effect on price.

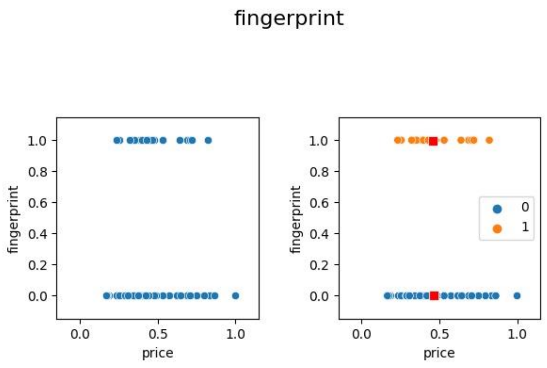


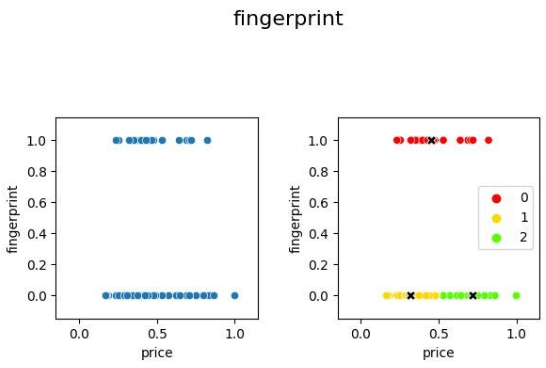


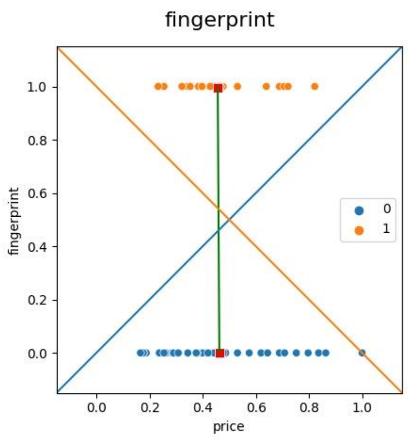


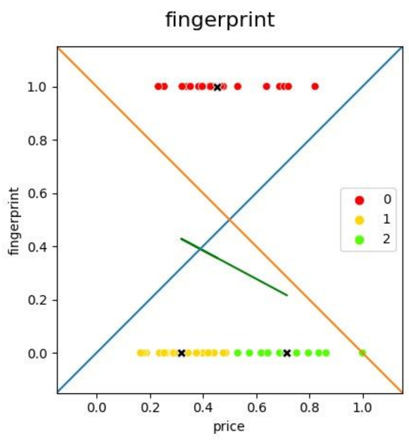


Fingerprint reader effect on price.

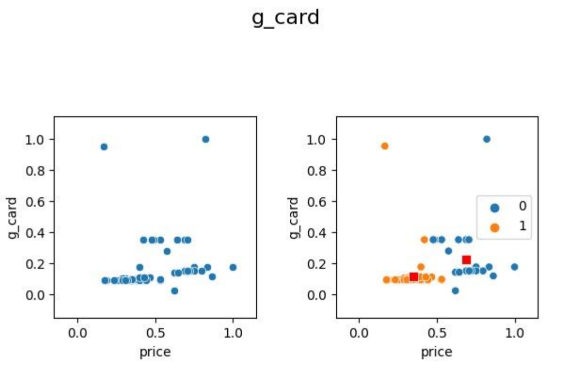


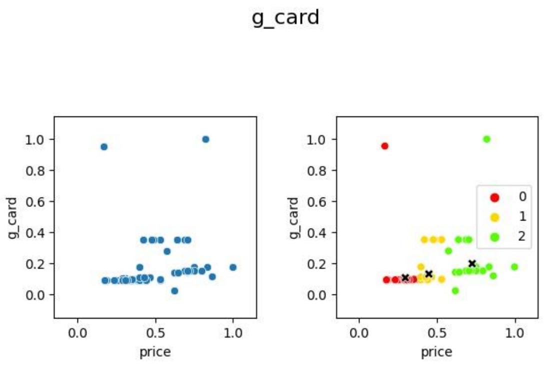


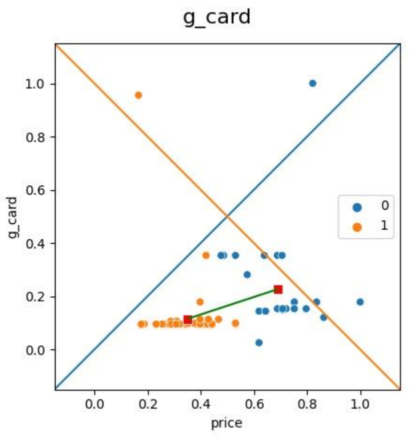


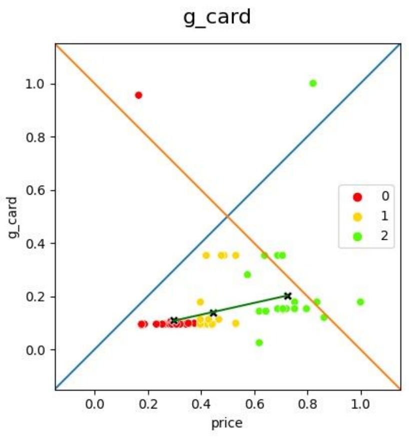


GPU effect on price.

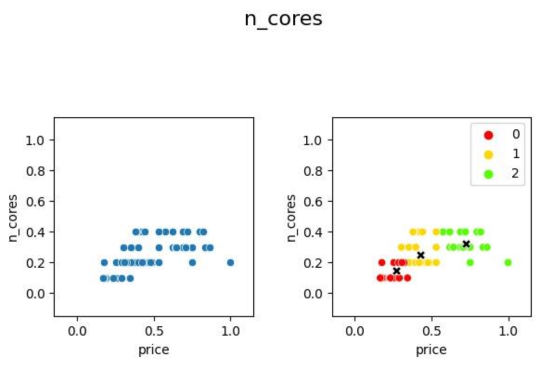


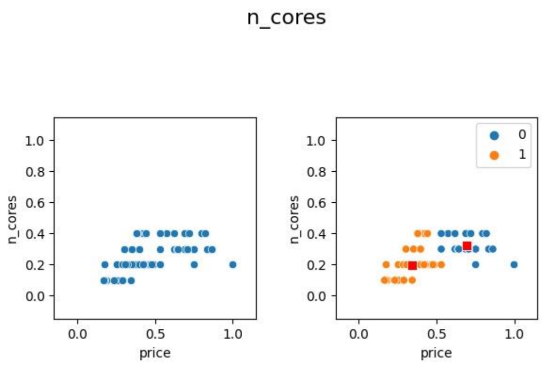


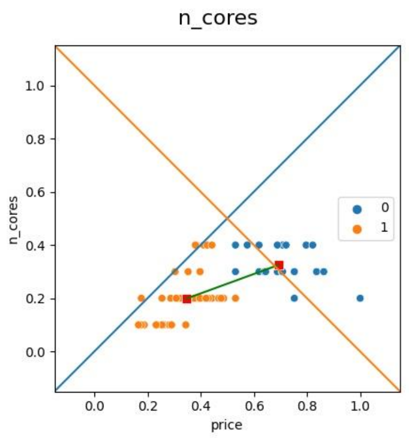


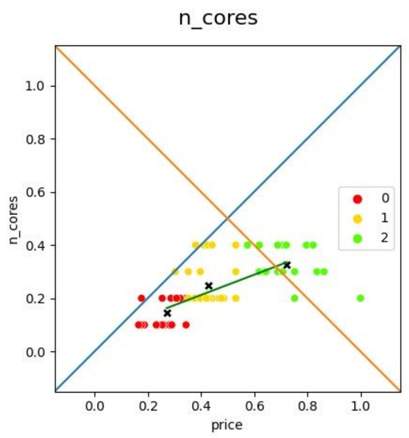


Number of CPU cores effect on price.

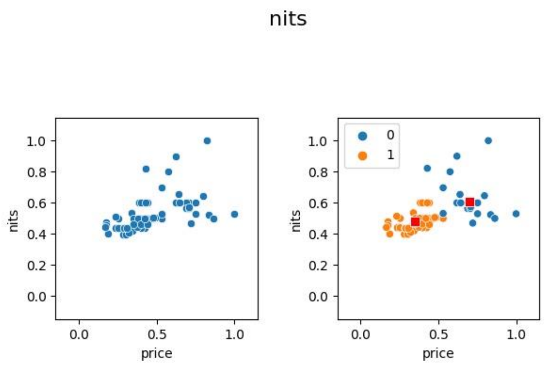


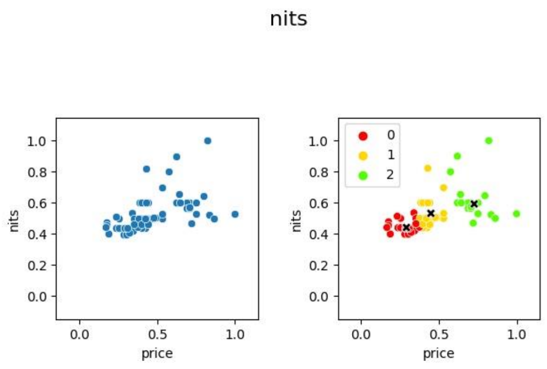


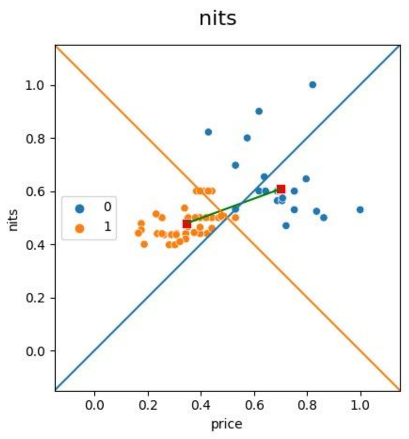


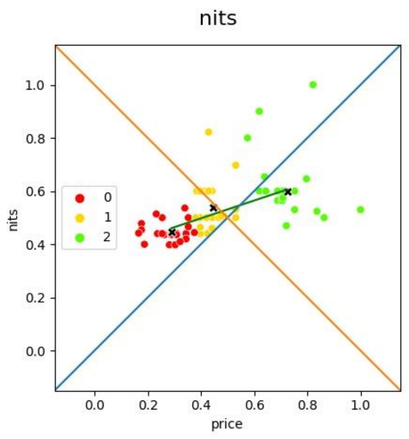


Brightnesss (nits) effect on price.

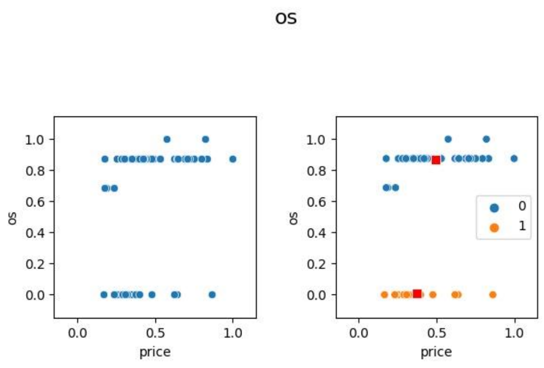


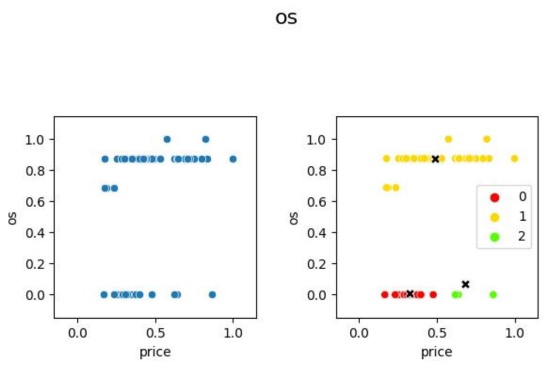


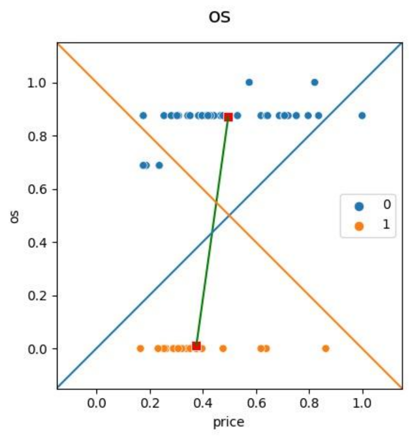


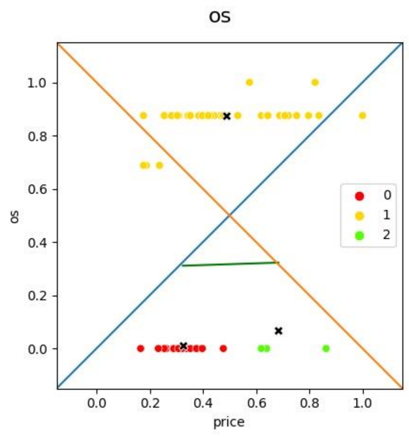


Operating system effect on price.

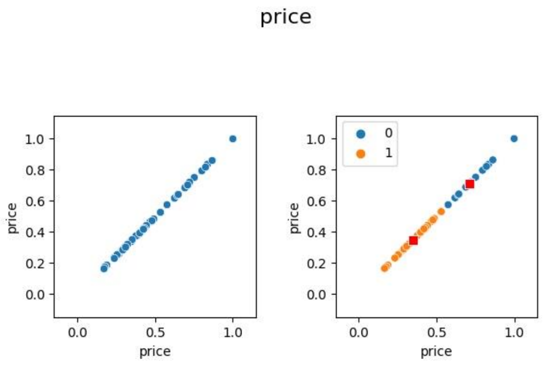


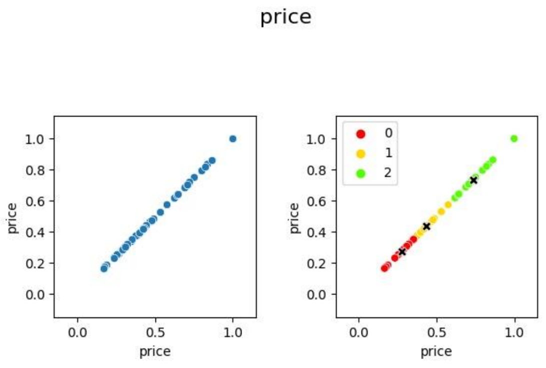


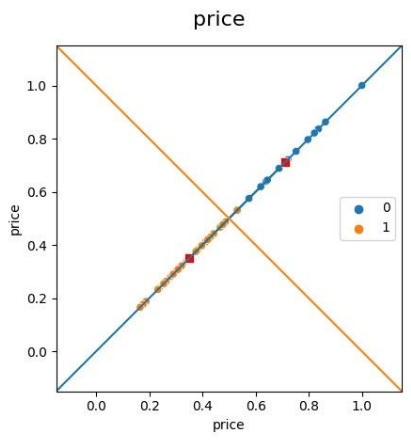


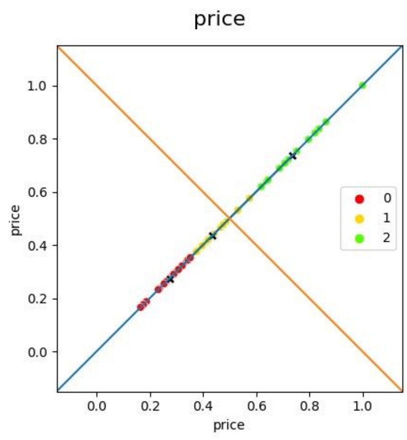


Price effect on price.

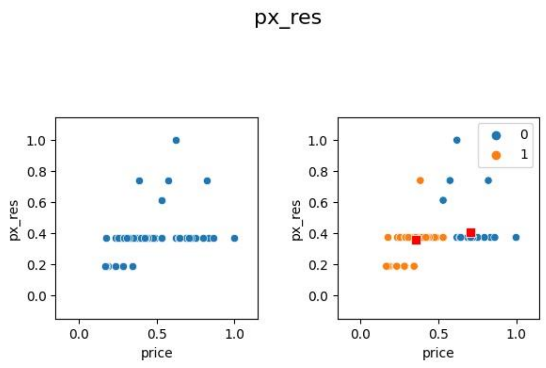


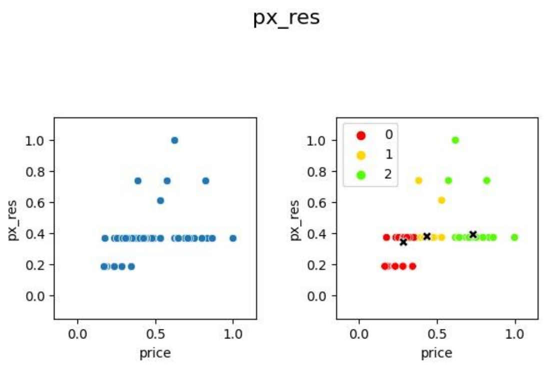


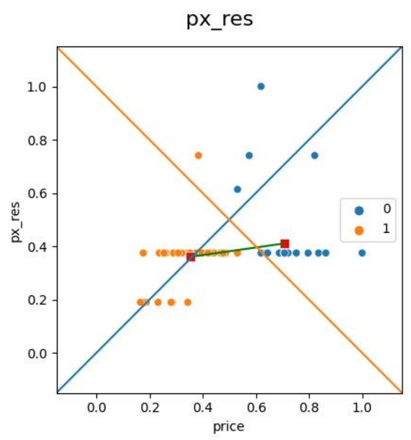


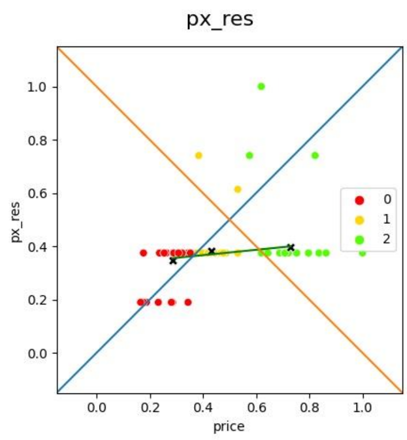


Resolution effect on price.

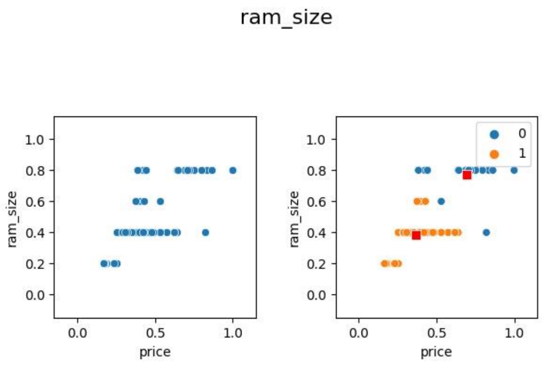


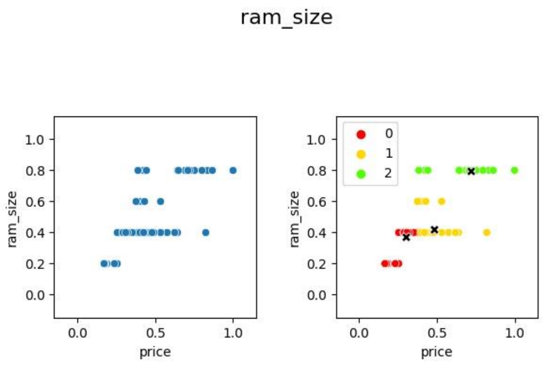


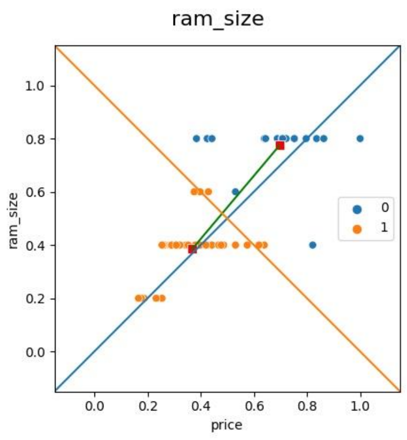


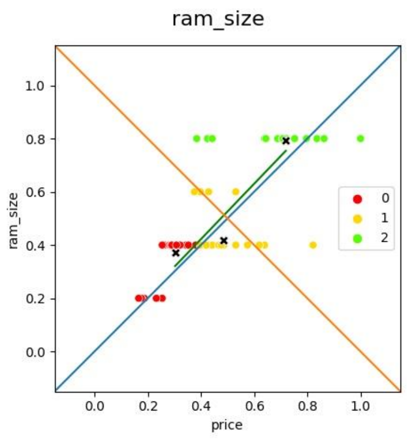


Ram size effect on price.

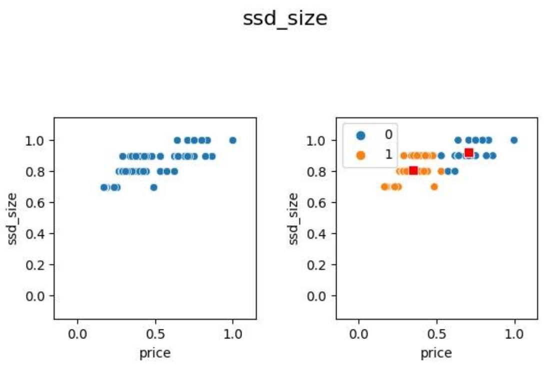


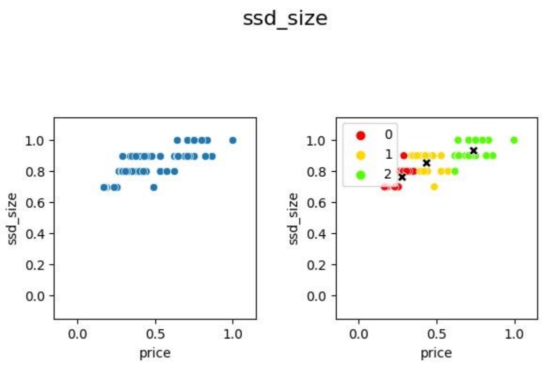


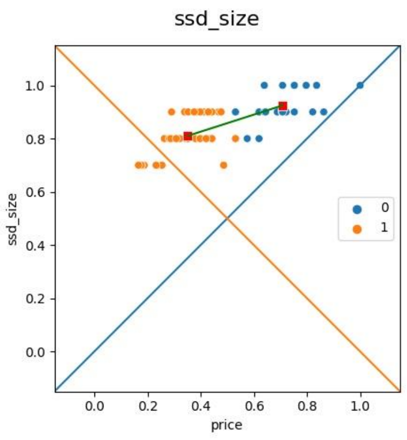


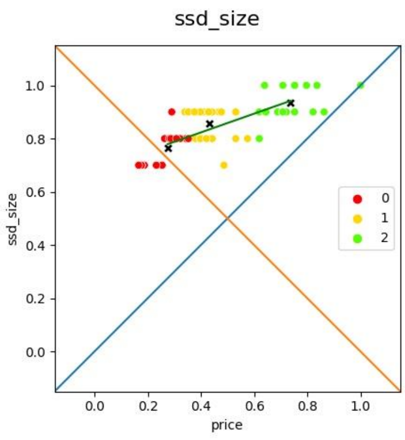


Solid state disk storage effect on price.

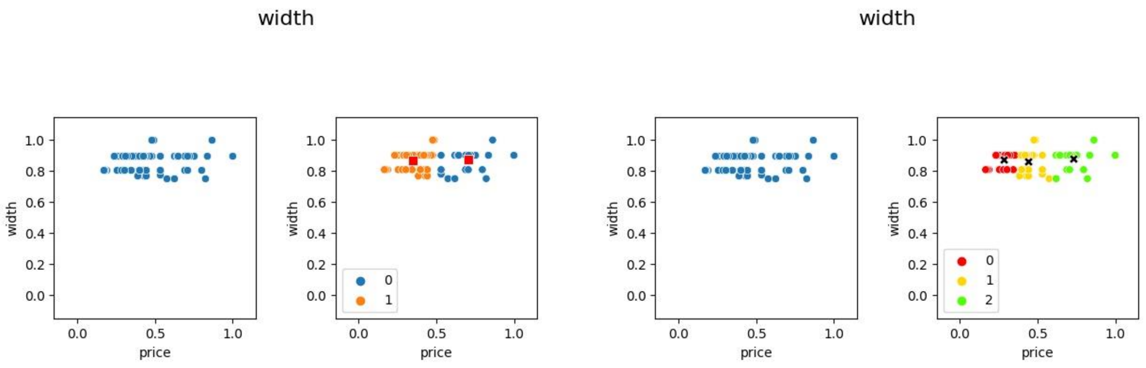


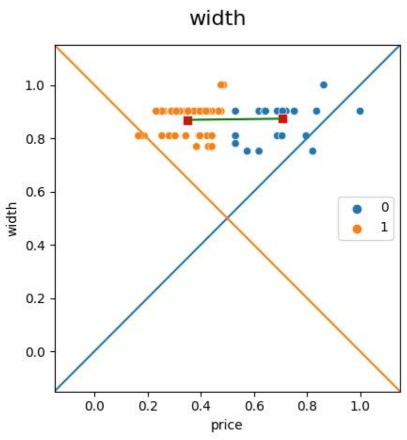


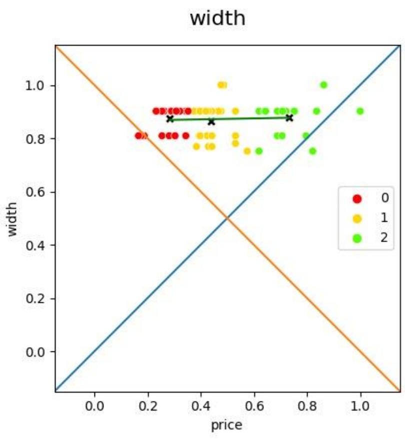




Screen width effect on price.


width
width
width
price


width
width
price


width
width
price