Master’s Thesis

Product Categorization for Customer Recommendation

From

Abhishek Ravindra Kurup

Date 05-03-2021

Department 3

Project Management and Data Science (MPMD)

1st Supervisor: Prof. Dr. Tilo Wendler

2nd Supervisor: Dr. Yousof Mardoukhi

Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

Restriction note

The present thesis with the title “Product Categorization for Customer Recommendation” contains in-house data of the company GG Brands Gmbh.

Therefore, it is only assigned for the HTW Berlin - University of Applied Sciences as well as the supervisors of the thesis. The thesis must not be made publicly available, neither made available to unauthorized third persons.

Berlin, 05-03-2021	[illegible]
(Location, date)	(genuine signature)

Index

List of figures

List of tables

List of code snippets

Index of abbreviations

RGB	Red, Green, Blue
CMYK	Cyan, Yellow, Magenta, Key (Black)
HSL	Hue, Saturation, Lightness
HSV	Hue, Saturation, Value
HVS	Human Visual System

1 Introduction

JewelCandle, now a part of GG Brands Gmbh, is a luxury scented candles producer founded in 2012 in Berlin, Germany. The concept started as a romantic gift idea in the founder's kitchen when he wanted to gift something handmade and unique to his partner. Since his partner loved scented candles, he thought of making one at home with the additional inclusion of a hidden piece of surprise jewellery. This idea gave birth to the business model of JewelCandle. The business focuses mainly on female clientele but it is not limited to the particular demographic, with some candles targeting the male audience as well. After starting in the German market, JewelCandle now has its presence around Europe and also in the U.S and India. The primary products of the company are scented candles with a piece of surprise jewellery inside and this concept has been spread to bath bombs and the seasonal Adventskalender as well. When a customer plans on buying one of the company's product, they can either visit one of the partner stores, the store website, or online marketplaces such as Amazon, eBay and C-Discount. While the customers cannot choose the design of the jewellery, they can either decide on the candle itself based on its looks and smell or the type of jewellery included within the candle. The customers can choose between ring, earring, bracelet, anklet types of jewellery.

When a customer decides to visit the company website to purchase a product they are provided with a list of available products and filters to help them shortlist their choices.

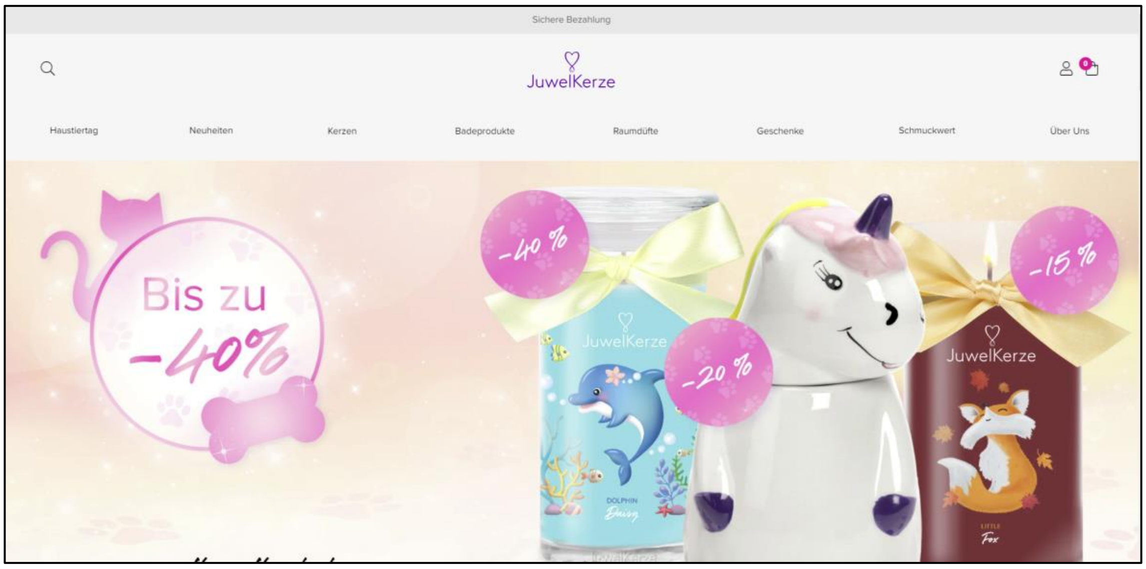
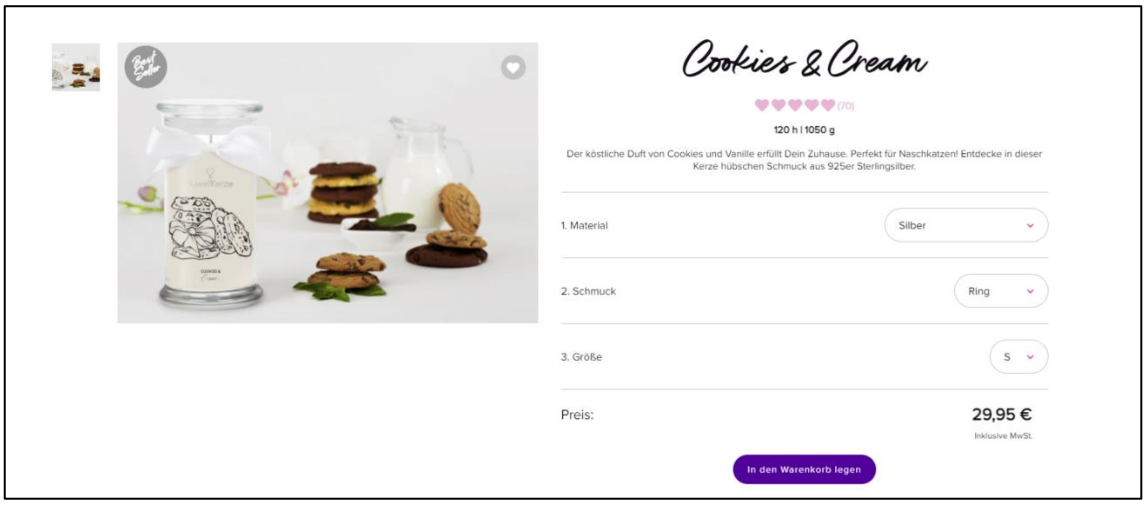


Figure 1: Home page of the company website for Germany.

Once the customer decides on a product, they can then visit the product page where they can choose other attributes of the product such as the size of the candle, the type of jewellery and the size of the jewellery. Along with these filters, the product page also includes a description of the product which usually includes the objective of the product, the different types of jewellery that a customer could potentially find within the product. Other information included on the product page is the fragrance, burn time, an option to provide a rating and review of the product and the delivery information of the product. Based on these factors a customer can make their decision to buy a certain product or not.



Cookies & Cream

(70)

120 h | 1050 g

Der köstliche Duft von Cookies und Vanille erfüllt Dein Zuhause. Perfekt für Naschkatzen! Entdecke in dieser Kerze hübschen Schmuck aus 925er Sterlingsilber.

1. Material: Silber

2. Schmuck: Ring

3. Größe: S

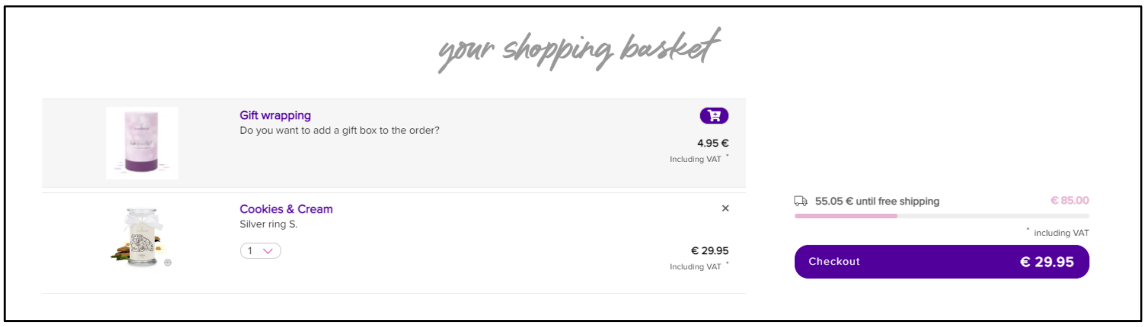
Preis: 29,95 €

Inklusive MwSt.

In den Warenkorb legen

Figure 2: Product page attribute selection

After the customer has decided on the product that they are interested in, they can add it to their cart and then either continue with shopping for other products or make the payment and complete their transaction. They also have the option to gift wrap the product if they intend to send it to someone else directly.



your shopping basket

Gift wrapping
Do you want to add a gift box to the order?
4.95 €
Including VAT *

Cookies & Cream
Silver ring S.
1
€ 29.95
Including VAT *

55.05 € until free shipping € 85.00

* including VAT

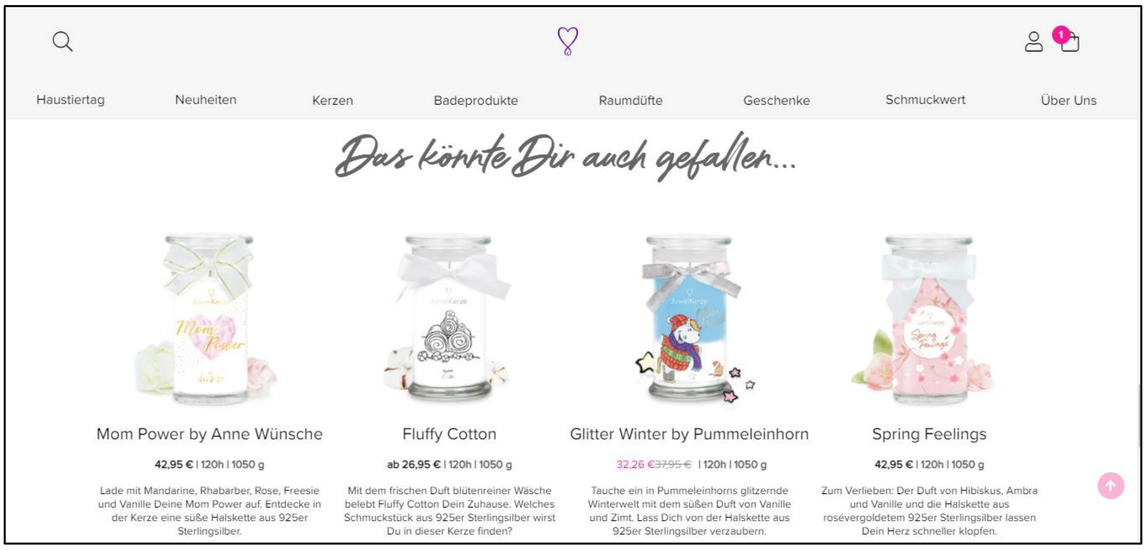
Checkout € 29.95

Figure 3: Order finalisation and check out

When a product is designed, a particular theme would be kept in mind and followed. An example would be a Halloween themed product such as the “Dia de Muertos” candle which was developed for that particular season. Other themes include summer and winter editions, collaborative products with social media influencers and many other festive occasions. And to promote all these products, some marketing activities have to be executed.

Figure 4: Halloween themed Dia de Muertos candle

These activities include running promotional sales, advertisements and creating popular social media content to expand the reach of the products. A customer usually visits the website or a shop after watching such promotions. Apart from direct marketing by the company, there would be instances of indirect marketing when a person sees a particular product at a friend’s house or in the form of recommendations from someone they know and they might only know the name of that particular candle. Following such influence, the potential customer searches for that product and eventually ends up on the product page. Since these promotions and advertisements are targeted towards a particular product there is always a possibility that a customer might not browse through the other products. Thus, it is desired to recommend other products while browsing. The website was recently migrated to the Shopify platform which has a recommendation system included that shows certain products that the customer “might also like”. However, the logic behind the recommendations is not clear and the recommended products are not necessarily related to the product the customer had initially selected.



Mom Power by Anne Wünsche

42,95 € | 120h | 1050 g

Lade mit Mandarine, Rhabarber, Rose, Freesie und Vanille Deine Mom Power auf. Entdecke in der Kerze eine süße Halskette aus 925er Sterlingsilber.

Fluffy Cotton

ab 26,95 € | 120h | 1050 g

Mit dem frischen Duft blütenreiner Wäsche belebt Fluffy Cotton Dein Zuhause. Welches Schmuckstück aus 925er Sterlingsilber wirst Du in dieser Kerze finden?

Glitter Winter by Pummeleinhorn

32,26 € ~~37,95 €~~ | 120h | 1050 g

Tauche ein in Pummeleinhorns glitzernde Winterwelt mit dem süßen Duft von Vanille und Zimt. Lass Dich von der Halskette aus 925er Sterlingsilber verzaubern.

Spring Feelings

42,95 € | 120h | 1050 g

Zum Verlieben: Der Duft von Hibiskus, Ambra und Vanille und die Halskette aus roségoldetem 925er Sterlingsilber lassen Dein Herz schneller klopfen.

Figure 5: Other recommended products on the "Cookies and Cream" candle product page.

Jewel Candle currently has many different candles in its product portfolio for many different occasions and as such there are some candles with similar colours and others with a different colour based on the theme and design. For example, summer edition candles usually have a lighter tone to give a feel of the sunny days and conversely, the winter editions have a darker tone. And special edition candles such as the Halloween edition shown in Figure 4, have a unique colour to remind the customer of the festival. These shades along with the perfumes act as stimulants for the moods of the customers to remind them of that particular theme. Keeping these details and the drawback of the current recommender system in mind, a recommender system that would be more efficient in creating recommendations based on the colour and the smell of the selected candle is being proposed. This new recommender system would be able to produce more personalised recommendations which are missing in the current implementation. The new “personalised” recommendations would be based on products having similar colours and smell since the fragrance of the products is one of its key attributes. This would start by identifying products having similar smells and similar colours independently. This thesis would however be focusing only on the similarity of colours between the products. This solution would then be merged with the solution to determine the similarity of smells in the future to form a single enhanced version of the generic recommender engine provided by the Shopify platform. Last but not the least the recommender system can be further enhanced by trailing each customer’s activities based on their historical purchases.

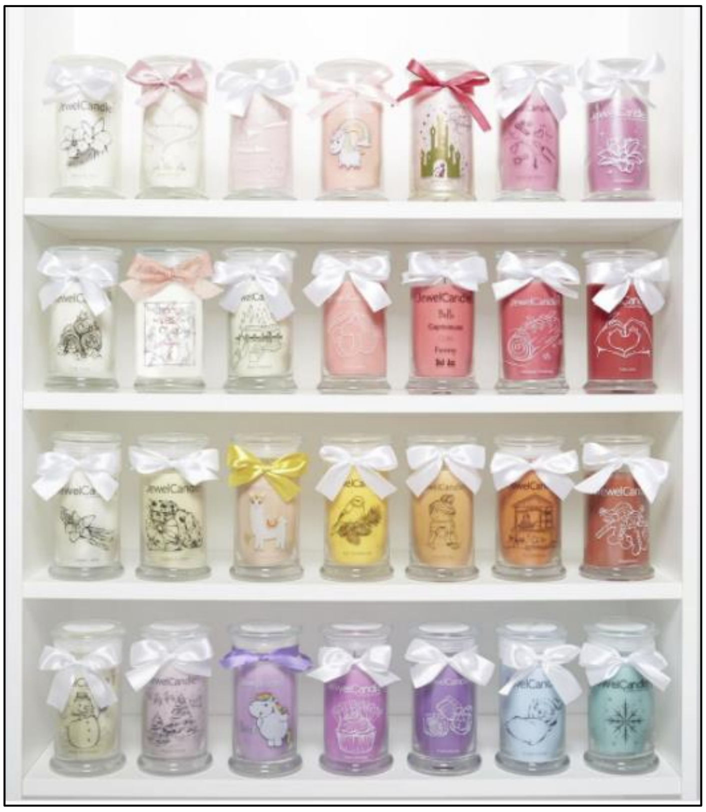


Figure 6: Different coloured candles in the product portfolio

While trying to identify products with similar colours it is important to understand how these colours might be perceived by the customers. Being a visual attribute there are a lot of dependencies that affect the perception. Some of the problems that come up are the differences in technologies used by the customer and also the biological and medical aspects of the human body. When it comes to technology, the quality of screen used in computer or phone displays vary depending on the model and price. For example, the colour reproduction on a display having 1080p or 4K resolutions would be much better than older technologies such as plasma displays and TFT screens with lower resolutions. And by taking biological and medical aspects such as colour blindness and poor vision into consideration it can be safely assumed that every human has a different perception of colour. Even if these factors are considered to be ideal, various other factors such as lighting, environment, device, etc. also make an impact. To reduce the effect of these drawbacks and to improve the experience of a customer, a generalised method is required to provide customers with recommendations of products similar to their initial selection. Keeping in mind the biological aspects of human vision, we can consider that, although powerful it still faces disadvantages depending on various factors. And a method to avoid the effects of external factors would be using an algorithm to perform image analysis.

To start working with an image analysis solution, it is important to start by understanding colours, the different formats of colours and which format would be more efficient and useful with this analysis. For example, RGB colour format is used as a base colour format for many photo editing tasks, CYMK is used for printing, HSL for computer vision etc. (Tkalcic and Tasic 2003). It is important to understand how the various colour models differ from each other to have a better idea of which model would work best for the problem being handled in this thesis. Some prominent colour formats along with their usage, advantages and disadvantages are discussed in more details in chapter 2. Based on one of these colour models, the next step would be to implement a form of image analysis. Apart from colours, it is also important to understand algorithms, since the combination of these aspects makes up the bases of image analysis. Image analysis has been used widely for various applications for example in the medical field to analyse biological conditions such as body temperature and other analyses such as identifying microscopic particles and distant planetary bodies (Fiete 2007). There have been researches conducted especially in the field of medical sciences where it has become important to segment different cells in the body with the help of colours (Bueno et al. 2008). These researches also include finding an appropriate colour model that works efficiently without the need for high computational power. Image analysis involves algorithmic methods to work with colours and images and thus is part of data science. It is, therefore, necessary to understand data science itself. The next section gives a brief introduction to Data Science and machine learning along with an overview of how these concepts can be used to solve this problem.

Using Data Science and Machine Learning as a solution

Although there is no one perfect definition of Data Science. But to put it simply, it is the study of data. It can be understood basically as an amalgamation of multiple disciplines such as statistics, mathematics and data analysis to gather useful information and insights by utilising various scientific processes and algorithms. Data science methodologies are applied to a large volume of data, also known as Big Data to perform complex calculations and predictions and also to build Machine Learning models (Aggarwal 2015).

When a person starts a Data Science project, some certain technologies and processes must be taken into consideration. Some of these include data mining, statistics, machine learning, Big Data, visualisations, data analysis and programming languages such as Python or R. These tools and skillsets enable a “Data Scientist” to solve complex data-related problems. Even having expertise over all these tools and skills, it is sometimes difficult to picture an appropriate structure or method to solve a problem. To solve this issue a standard process for data mining was introduced in 1996 known as the CRISP-DM (CRross Industry Standard Process for Data Mining) model (Rüdiger Wirth and Jochen Hipp 2000). This process was developed by leading data mining companies, DaimlerChrysler AG, SPSS, NCR, and OHRA with the sponsorship of the European Commission under the ESPRIT program. The CRISP-DM model is a guideline for the design and implementation of data mining and data science projects. While executing a data mining or a data science project there are various steps involved which begins with understanding the business requirement, understanding the available data set, preparing the available data set as required, creating a machine learning model for the analysis, evaluating the results from the model and finally deployment of the model if all the tests generate satisfactory results. During this process, there will be instances where it might be necessary to rethink strategies for modelling based on the evaluation of the model performance. The diagram below shows the CRISP-DM model and the flow of each of the steps involved.

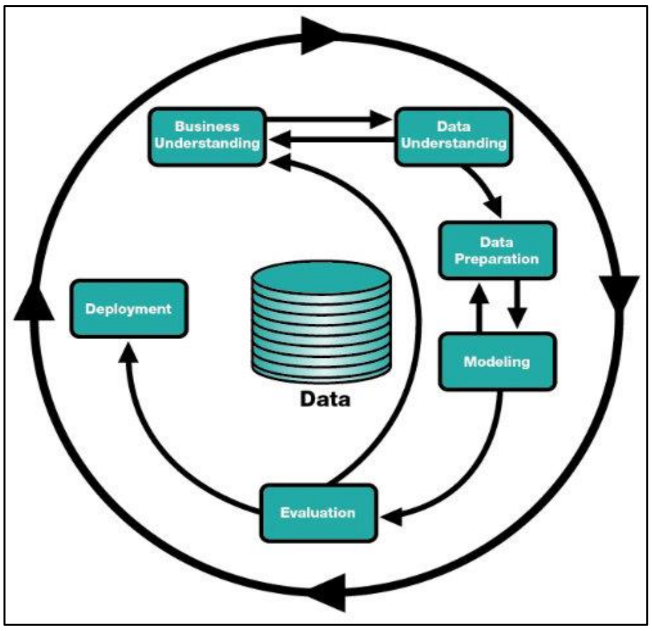


Figure 7: CRISP-DM Model describing the workflow of the data mining process (Nicholas Njiru and Elisha Opiyo 2018)

While the “Data Scientist” can gather the business requirements and the data set from the stakeholders, they have to identify the type of modelling and the appropriate data preparation they would require for their particular problem. There are a few different types of machine learning algorithms that are available to use, but before understanding the different types of algorithms, a general understanding of machine learning as a whole is required. The concept of machine learning was developed as a means to enable computers to improve their processes automatically without external influence. Newer technologies have enabled faster computation speeds thus resulting in faster results of complex calculations and finally progressing an old pipeline dream of curious scientists towards a widely utilised commercial process and further into artificial intelligence.

"*Machine learning can be broadly defined as computational methods using experience to improve performance or to make accurate predictions*" (Mohri et al. 2018).

The experience mentioned in the definition is the information already available related to the dataset based on which the model can make a decision. The dataset fed into the algorithms can be labelled or unlabelled, which means that the dataset might already contain information which the machine learning algorithm can use to base its prediction on, or such information might not be available in which case the algorithm would not have any help in generating these predictions. The larger the dataset, the better the efficiency and accuracy would be of the model, since it would have a lot of data points to base its decision.

There are three basic types of machine learning algorithms.

1. Supervised Machine Learning –

   These types of algorithms use labelled data to train themselves. In such instances, the available dataset would be split into 2 parts, namely the training dataset and the testing dataset in a set ratio. The model is trained using the training data and predictions are made for the test data. These types of algorithms are used for classification and regression problems. Some examples of the algorithms under supervised machine learning are logistic regression, naïve Bayes, decision trees and support vector machines. When using this type of model, care must be taken that the training dataset is properly labelled, since the final predictions of any other input dataset depend on the training of the model.

   The learning process of a typical supervised machine learning model consists of the below basic components.

- Labelled dataset split into training and testing datasets. The train-test split ratio is based on the use case and is typically around 70:30 or 80:20, where the larger number is the training dataset.
- The algorithm itself, which would be finalised after testing multiple algorithms to see which model provides the best result. The training dataset is used to create the different models and the training dataset is used to test the performance of each model.
- An evaluation dataset, which is usually different from the training and testing dataset, is finally used to test the algorithms and finalize the model.

The image below shows the workflow of the learning process of a supervised machine learning model.

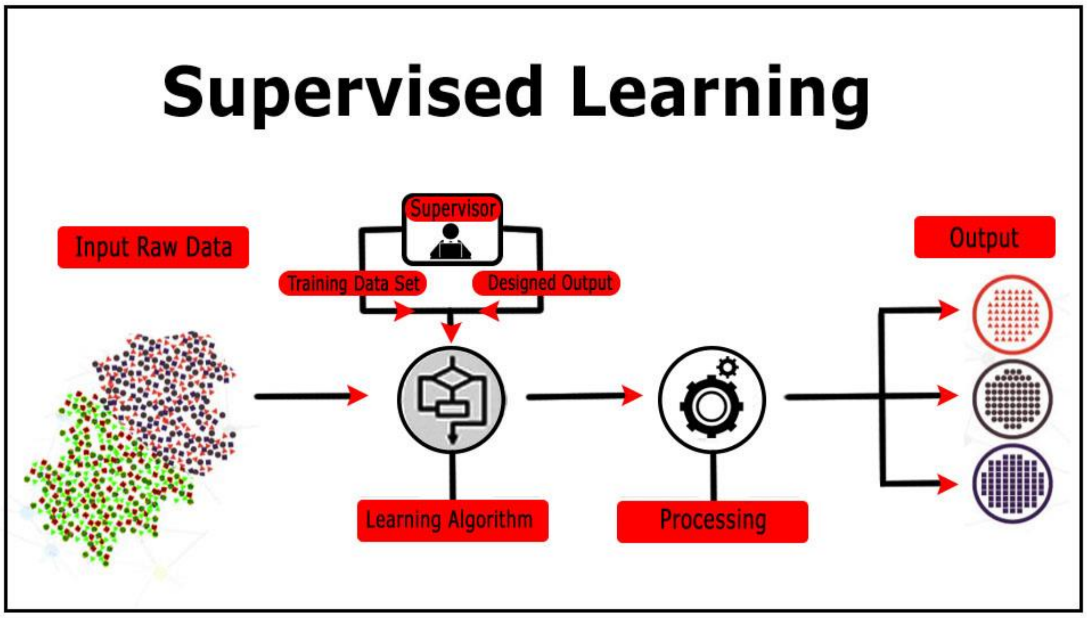


Figure 8: Supervised Machine Learning process (SUPERVISED MODELS | Data Vedas 2021)

Some example use cases of supervised machine learning are the prediction of power usage in a particular area based on previous usages in the same period and another use case would be to predict which account holder in a bank has the possibility to default on a loan.

2. Unsupervised Machine Learning –

   There will be situations where the labelling of data would not be available and it would be very difficult to manually generate these labels. To handle this drawback of the supervised machine learning algorithms, the concept of unsupervised algorithms was developed. Unlike supervised machine learning models, this type of model does not require a labelled dataset. The dataset is directly fed into the model and the algorithm presents the output based on the set parameters. These algorithms are used in situations where the dataset needs to be clustered in groups or to label when the dataset is not labelled. Some example use cases of the usage of these algorithms are categorisation, customer segmentation, developing recommender systems and fraud detection in the financial sector. Since the use case of this thesis is to analyse colours and develop clusters of products based on similar colours, this thesis would make use of unsupervised machine learning. Another reason to use this type of algorithm is that even though the attributes of the product have a colour value, it cannot be used as an identifier for similarity. Further chapters discuss the various models that were tested for this use case. The image below shows the workflow of the learning process of an unsupervised machine learning model.

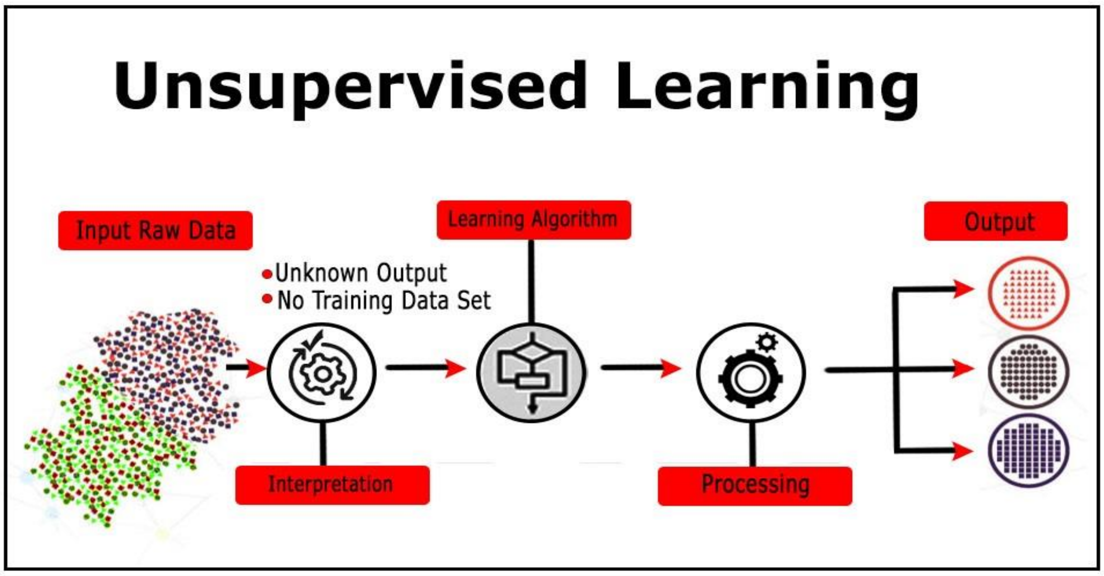


Figure 9: Unsupervised Machine Learning process (UNSUPERVISED MODELS | Data Vedas 2021)

3. Reinforcement Machine Learning –

   Reinforcement learning is the process of enabling a program to take actions based on the situations it is put through to increase the reward received. In this case, the model (agent) continuously learns starting with an unknown dataset and then starts predicting the instances occurring in its environment. The model actively interacts with its surrounding environment and learns by receiving a reward or punishment for each action it undertakes. This method is generally used to find the best solution for a problem. Some examples of reinforcement learning being used are a chess game or a more advanced use case is a self-driving car. An example of an achievement of reinforcement learning was in October 2015, when an algorithm with the help of deep neural networks managed to defeat the three-time European champion Mr Fan Hui with a score of 5-0 in the complex strategic game of Go. This program was developed by DeepMind (now owned by Google) and is known as AlphaGo. This program has been awarded a 9 dan professional ranking which is the highest professional certification in the game (Deepmind 2021). The figure below shows the learning process of a reinforcement model.

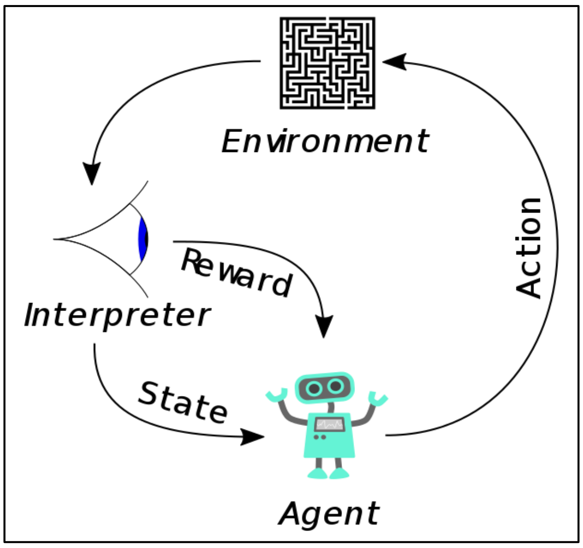


Figure 10: Reinforcement machine learning process (Wikipedia 2021b)

After achieving a basic understanding of data science and machine learning concepts, it can be understood that the problem of identifying similar colours is a clustering problem and can be solved with the help of unsupervised machine learning algorithms. We will be testing some mathematical models along with a simple function using comparative logic and some unsupervised machine learning models such as K-means clustering and DBSCAN algorithms to see which model yields a better result and would finally be a better fit for this proof of concept.

To summarise, the general objective of this thesis would be to test different approaches and finalise a model to produce groups or clusters of similar looking products based on their colours. These clusters form the initial step towards the building of an improved recommender engine that utilises the key attributes of the products i.e., the colour and smell. Following this understanding, the questions that need to be answered are,

- Which colour model would be the most efficient for our use case?
- For that particular colour model what kind of algorithms would work best?
- Would there be a need to define or develop an independent logic specific to this use case?
- And finally, how can the model be evaluated?

Following the introduction of this project, the thesis is structured as follows. Chapter 2 describes briefly the human visual system and how humans generally perceive different colours. Along with the visual system, the chapter also describes the concept of colours, the different "Colour Spaces" and how each model is different from the other and eventually decide on a colour model to start our proof of concept. Chapter 3 focuses on the available data set that is used in this thesis. It also describes the data preparation steps that were taken to form the basic structure which would be fed into different clustering models. Chapter 4 describes the different functions and clustering algorithms that were considered as possible approaches to solving our problem. Chapter 5 determines the model with the best results based on possible visual and quantitative tests for clustering algorithms. Chapter 6 goes through the conclusion and future scope and expansion of this project.

2 Basic understanding of human vision, the concept of colours and colour spaces

To answer the first question of which colour model would work best for our use case, it is necessary to first understand how the human visual system works. This plays an important part in understanding how we as humans perceive colours and how different colour models were developed to understand this perception better. These models also have various usages depending on the complexity and feasibility of implementation. The next section starts by describing the part of the human eye that has been discovered to be responsible for perceiving colours. This part plays a crucial role for human vision and deficiencies of this part of the human eye can cause vision disorders such as colour blindness. Since vision deficiencies like colour blindness would also affect a customer's experience during their purchases on the website this gives us more of reason to improve the recommendations and the overall experience of the customer's visit.

2.1 The human visual system

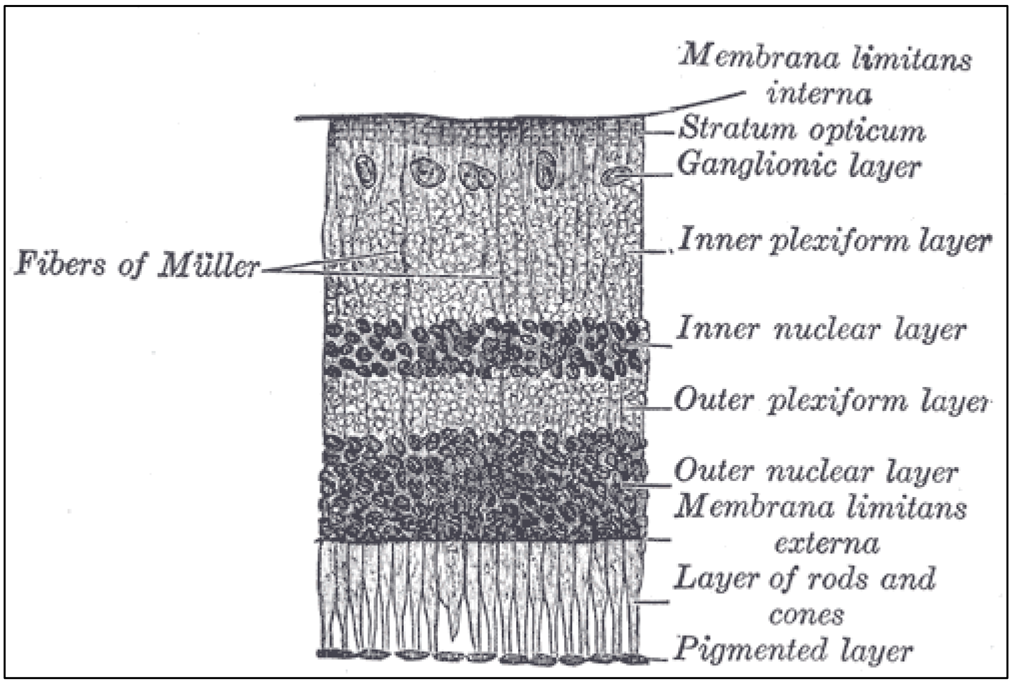


Figure 11: Section of Retina (Gray 1878)

The retina of the human eye consists of multiple layers as shown in the image above. Out of these layers, the rods and cones are responsible for the identification of colours. Rods are highly sensitive to light and cones are less sensitive thus rods being better in low light levels. While rods are efficient with detecting the intensity of light, cones help in detecting the saturation and colours. These rods and cones are the photoreceptors of the human eye. The fovea is another part of the human eye that is responsible for a sharp vision. Different theories have been presented in regards to human perception of colours such as the “Trichromatic Theory” developed by Thomas Young and Hermann von Helmholtz (Helmholtz 2013), which says that humans primarily perceive colours with 3 main colours and their combinations i.e., Red, Blue and Green. Another theory known as the “Opponent Process Theory” developed by Ewald Hering (Buchsbaum and Gottschalk 1983) suggests that human perception is more antagonistic in nature. This means that humans see colours more as opposing colour pairs such as red vs green, blue vs yellow, and black vs white. Even if we consider that humans have a Trichromatic vision, it would mean humans can perceive around 10 million colours, but it again differs from person to person depending on their health and eye conditions such as colour blindness. The colour vision of humans is many times also affected due to the nature of their jobs, diseases or old age. Apart from biological reasons, differences in colour perceptions extend towards the different forms of media it is portrayed on to, for examples digital media such as websites or print media such as magazines. These discrepancies are caused due to the limitations of technology where a monitor or a printer might not be able to accurately present a particular colour and also because technology is not standardised and the quality depends on the price paid for that particular piece of equipment.

Now that there is a brief understanding of how human vision functions when it comes to the perception of colours, the next step is to understand the concept of colours itself. Even though we are always surrounded by colours, we hardly think about it and how it could be described other than using adjectives such as “bright”, “dark” and “fun” among many others. The next section will go deeper into the concept of colours by expanding into their attributes such as hue, saturation, lightness etc.

2.2 Understanding the concept of colours

"*Color is the visual sensation, associated with a part of the field of view that appears to the eye to be without structure, through which this part can be distinguished from another unstructured neighbouring area when observed with a single, unmoving eye*" (German Institute for Standardization 2017).

The visual sensations described in the definition above can be categorised into a few broad types of sensations.

- Brightness / Lightness – The sensation to perceive the quantity of light exhibited by an object. In other words, how bright or dark an object is.
- Hue – The sensation to understand the difference between shades of colours and perceive how similar or different one colour is to another. For example, where colour is blue, green or somewhat similar to red.
- Saturation / Colourfulness / Chroma – The sensation to perceive the intensity of the hue exhibited by an object.

From the above-mentioned sensations, the hue, saturation and brightness are the usual attributes used to qualitatively describe a colour. The image below gives a brief understanding of these attributes.

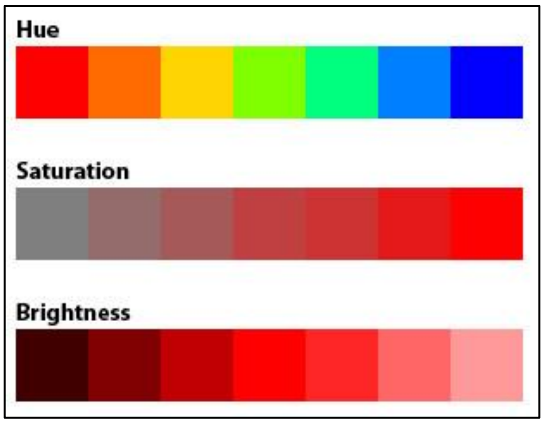


Figure 12: Hue, Saturation and Brightness

The different categories of colours are identified by the wavelength of light when it is reflected off of the object. These reflections depend on various properties of the object such as absorption, refraction, etc. The perception of colour also depends on receptors. Due to these reasons, the perception of colours is extremely subjective. It is difficult to assign numerical values to visual stimulation. This is where colour spaces come into the picture. The goal of colour spaces is to assist the measurement and to describe colours between people and/or machines or programs (Cotton 1995).

2.3 Understanding different colour spaces

A colour space is a systematic organisation of colours by assigning colour names or numbers or represented mathematically in the form of tuples which helps with visualising the colours. When represented in mathematical form, colour spaces can also be known as colour models. Colour models are usually represented in the form of a tuple of three or four numerical values which represent certain properties of that colour.

Different types of colour models were developed while keeping certain applications for them in mind. For example, colours displayed on a monitor is usually a form of the RGB colour space. This is because, in the early days of Cathode Ray Tube (CRT) monitors, phosphorous emissions of Red, Blue and Green were used to create colour images (Display device and cathode ray tube 2003).

The image below shows how colours were displayed in a CRT Monitor.

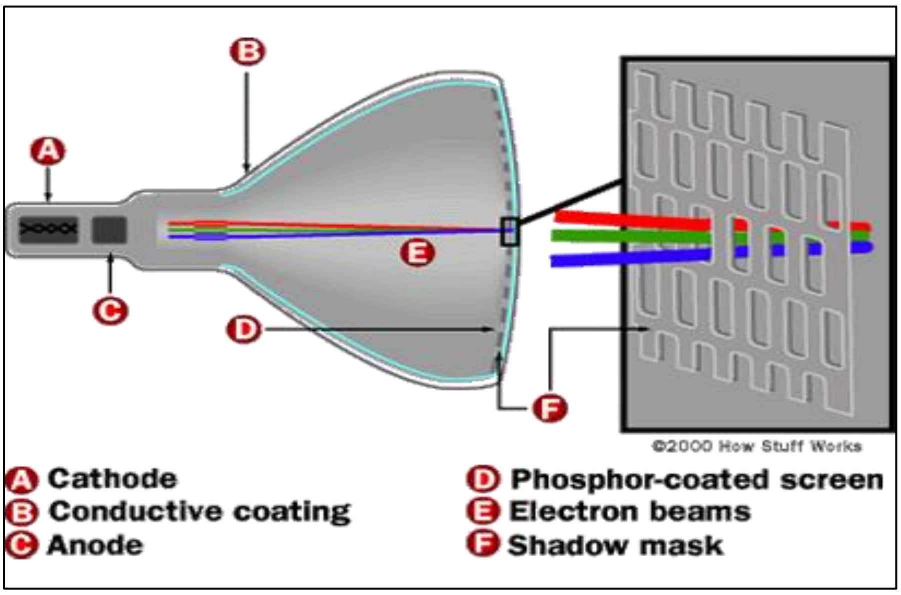


Figure 13: Working of CRT Monitor (How Computer Monitors Work 2000)

On the other hand, printing presses use the CMYK colour space as using a combination of Cyan, Yellow and Magenta along with Key (Black) can help in producing a wide range of colours for print media. The reason why RGB is not used for printing is that RGB colours can only be viewed in natural or artificial light, which means that there has to be an external light source to view the colours properly. CMYK is a subtractive combination of RGB and can be viewed easily on paper (Tkalcic and Tasic 2003). This also causes a difference in colours as the same colour is perceived differently in both spaces. Graphic designers usually create their designs in the RGB colour space and then convert them into CMYK space for printing. The CMYK model functions by covering up or masking the primary colour partly or wholly by making use of a lighter background, e.g., white paper. This model is also known as a subtractive model because the ink reduces the reflection of the light by the colour. The white light reduces the amount of red, green and blue colours. These simple transformations provide results only good enough for printing purposes and for any other purposes, the other colour models are usually preferred. It is also because of these simple conversions that an image on paper rarely matches the image displayed on the screen.

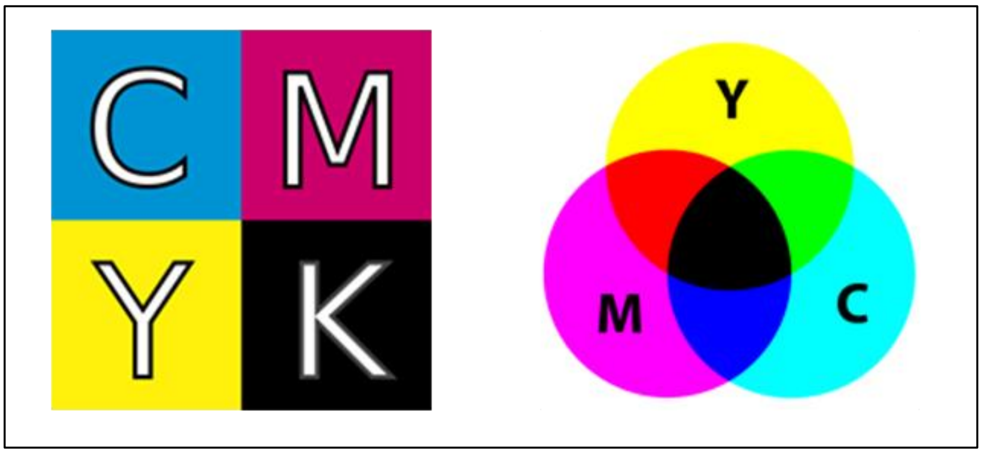


Figure 14: A representation of the CMYK colour combination (Wikipedia 2020a)

There is also the case of some colour spaces being perceptually linear in nature. This means that any amount of change in stimulus will have the same change reflected in its perception. While on the other hand, some colour spaces especially the ones used in computer graphics are non-linear in nature. There are also some colour spaces where the user can use them very intuitively and create their own blend relatively easily. And some spaces can be a little confusing for the user to work with. And finally, there are some colour spaces that are device dependent and others that work equally well no matter what environment we use them in. In other words, such colour spaces are device-independent.

The different colour spaces that are directly or indirectly used or considered in this thesis are explained in the further sections.

2.3.1 RGB Colour Space

RGB (Red, Green, Blue) is a colour system that is based on the “Trichromatic Theory” developed by Thomas Young and Hermann von Helmholtz (Helmholtz 2013). The idea behind the development of the RGB colour space was to try and simulate the perception of human vision. Human eyes constitute three types of cones that respond to different wavelengths of the light spectrum.

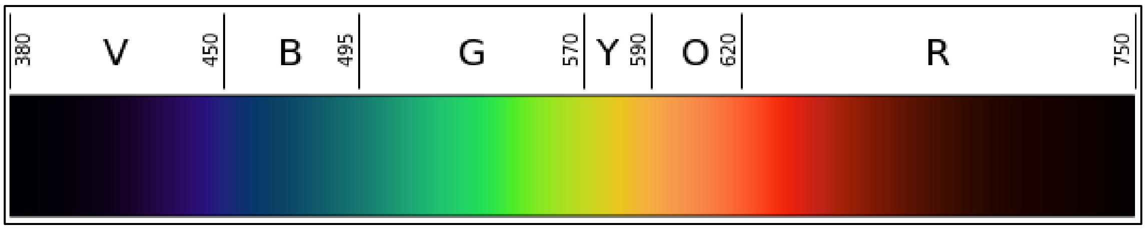


Figure 15: Human Visible light spectrum (nm scale)

These cones are usually referred to as Red, Green and Blue which approximately represent Long, Medium and Short wavelengths perception (Helmholtz 2013). As described previously, RGB is dependent on an external light source for an accurate representation of the colours. Hence, the RGB colour space is also known as device-dependent.

By adding Red, Green and Blue in varying quantities we are able to create various other colours. This property of the model makes it a type of additive colour model. Additive colour is a property of a colour model that produces a different shade of colour or rather the shade can be predicted when the numerical values of the primary colours are added.

Colours in the RGB colour space is represented numerically by indicating the amount of Red, Green or Blue present in that colour. These numerical values range from (0,0,0) which represents the colour Black to (255,255,255) which represents white. 0 is the lowest possible value for colour while 255 being the highest.

Since colours are defined by their 3 properties in this model, the 3 numerical values are considered as Cartesian Coordinates in Euclidean space as a three-dimensional geometric figure. In the case of RGB, it is depicted in the form of a cube as shown in the figure 16. The values represented by the cube are non-negative and lie in the range of 0-1. The point of origin (0,0,0) is assigned to the colour Black and the values keep increasing till the brightest point (1,1,1) which is assigned to the colour White.

In the RGB colour model, Red is projected on the horizontal X-axis with its values increasing to the left. The colour Blue is projected on the Y-axis increasing in values from left to right and Green projected on to the Z-axis in the increasing order from bottom to top. The origin point, the colour Black is on the other side of the cube.

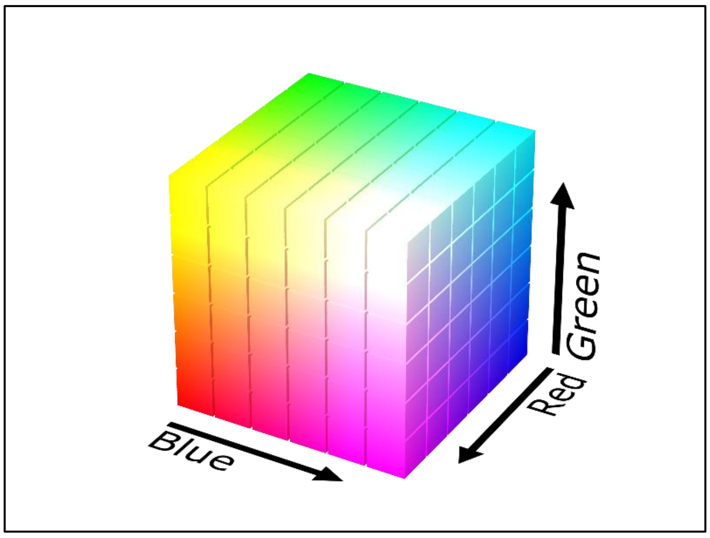


Figure 16: RGB Colour Space

There are a few shortcomings of the RGB model that affects this analysis. The main disadvantage is the low correlation between two colours and their Euclidean distance in the RGB space (Tkalcic and Tasic 2003). This means that even if the Euclidean distance calculated between two colours seems quite low, the actual perceived colour might be quite different.

2.3.2 HSL/HSV Colour Space

The HSL (Hue, Saturation, Lightness) colour space is a linear transformation of the RGB colour space. This colour space is also known as HSV (Hue, Saturation, Value), HIS (Hue, Saturation, Intensity), HCI (Hue, Chroma, Intensity), etc. Since the HSL colour space is derived from the RGB space, it also inherits the device-dependent and non-linear properties of the RGB space.

The HSL/HSV space was designed by computer graphics researchers to make it more similar to the human perception of colours when compared to the RGB model. The HSV hexcone model was proposed by Alvy Ray Smith in 1978 (Smith 1978), as an alternative to the RGB monitor gamut which is computationally much slower. This was done by arranging the primary hues in a cylindrical manner where the Red primary is placed at 0°, the Green primary at 120° and the Blue primary at 240° and then continuing back to Red primary. The central axis consists of achromatic or grey colours ranging from White (lightness, value=1) at the top to Black (lightness, value=0) at the bottom. The lightness or value is denoted as the percentage value of the lightness of the colour. The development of this colour space was motivated by the need for adding colour encoding to the existing television broadcasting system in 1938 which was monochrome. This development helped remove the need for modifying the existing receivers and facilitated the reception of colour broadcasts.

The image below shows the arrangement of colours as per the hues along with their lightness and the HSL values of each colour as a sample.

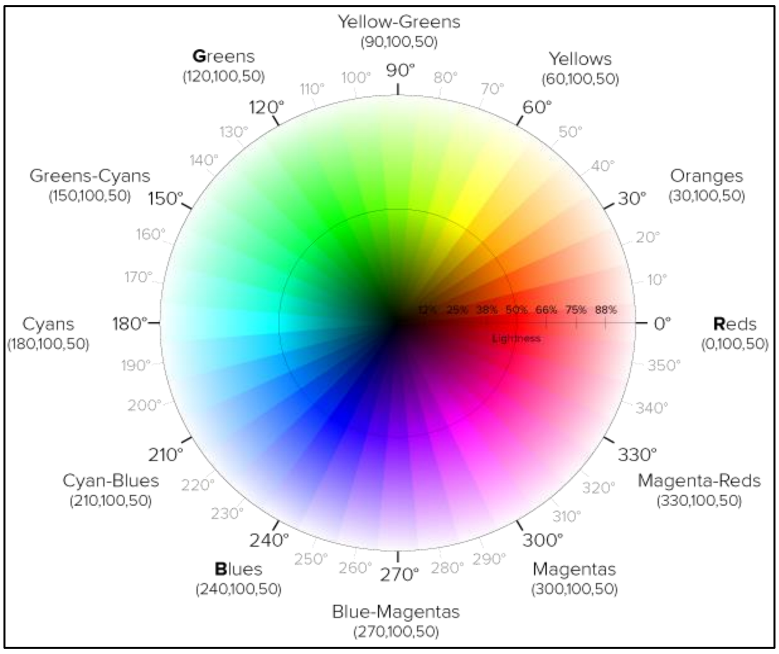


Figure 17: HSL colour wheel (Codrops 2016)

The images below show the cylindrical HSL/HSV models and when they are projected using chroma values.

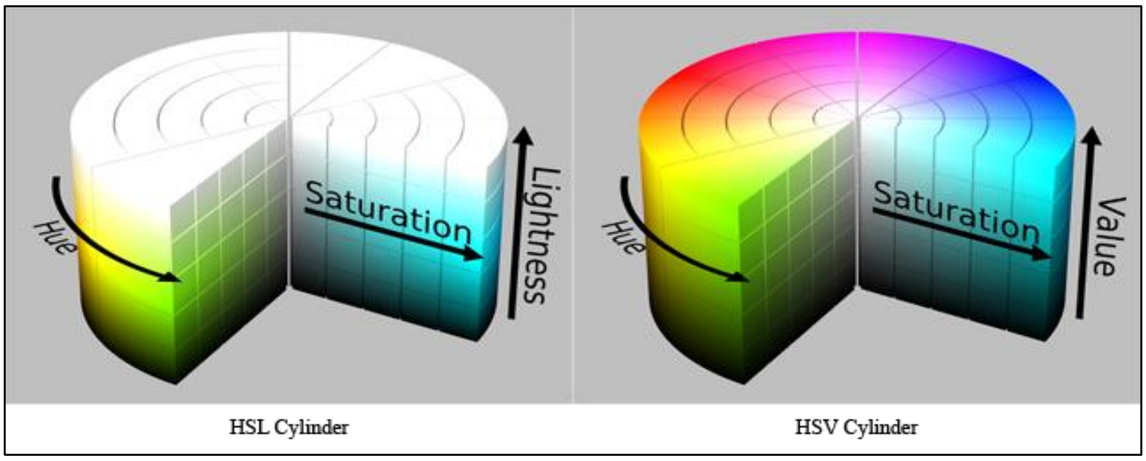


Figure 18: Representation of HSL and HSV models in the cylindrical form (Wikipedia 2020b)

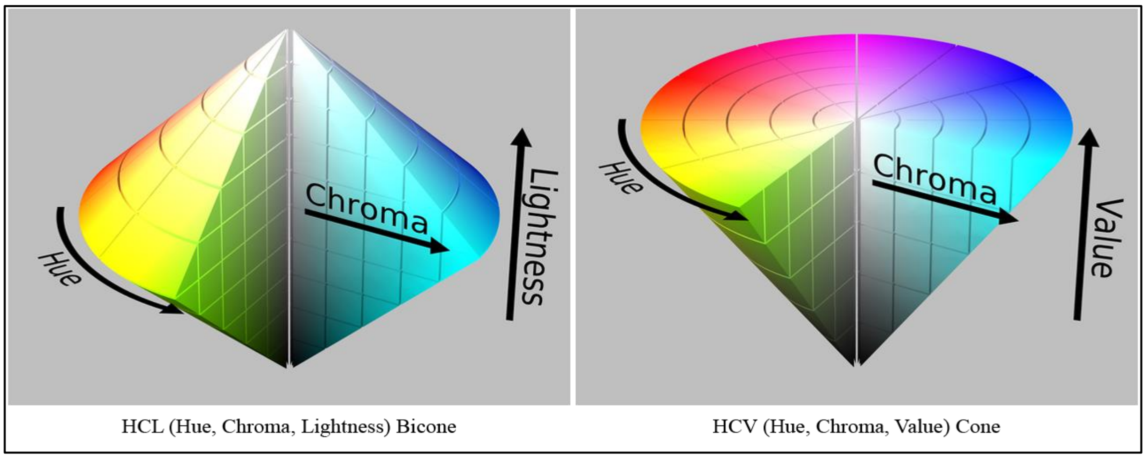

Chroma
Lightness
Hue
HCL (Hue, Chroma, Lightness) Bicone
Chroma
Value
Hue
HCV (Hue, Chroma, Value) Cone

Figure 19: Bicone format of HCL model and the conical format of HCV model (Wikipedia 2020b)]

The HSL/HSV is frequently used in image analysis and computer vision. Some examples of this are the identification of license plates, medical image analysis, face recognition etc. This is because when it comes to computer vision, algorithms do not have to strictly follow the Human Visual System. Just being within a certain range of the Hue, Saturation, Chroma or lightness can be effective in such analysis.

2.3.2.1 Conversion of RGB to HSL values

Conversion of RGB values of a colour to its HSL values is done based on the following steps.

Step 1: Convert RGB values to the range of 0 to 1. This is done by dividing the individual attribute values by 255.
Step 2: Check the minimum and the maximum values between R, G and B.
Step 3: Calculate the Luminance of the colour by finding the average of the min and max values, i.e., add the min and max values and divide by 2.
Step 4: Calculate the saturation. The saturation depends on the min and max values. If the min and max values are the same, that means that the colour is a shade of grey. Depending on the brightness, the colour might lie somewhere between black and white. If there is no saturation, then the need for calculating the hue is eliminated and it is set as 0.
Step 5: Once we have determined the presence of saturation, the further calculation is dependent on the value of the luminance.

If luminance < 0.5, then Saturation = (max - min) / (max + min)

If luminance > 0.5, then Saturation = (max - min) / (2.0 – max - min)

Step 6: Finally, to calculate the Hue, we need to check which attribute of the RGB values is the greatest in value. The formula for Hue is dependent on the attribute having the max value.

If Red is max, then Hue = (G – B) / (max – min)

If Green is max, then Hue = 2.0 + (B - R) / (max - min)

If Blue is max, then Hue = 4.0 + (R - G) / (max - min)

Next, we convert Hue to degrees by multiplying the calculated Hue value by 60. In case the value of Hue is negative, we need to add 360 to it since the Hue wheel is circular in shape.

Even though the HSL model can be used to more or less cover the range of colours seen by the human eye, there are some shortcomings as well. One of the flaws is that, though this model performs very well in image analysis and is much closer to human perception of colours compared to the RGB model, it is still far from actual human perception. The CIE colour space was developed keeping this in mind and to represent colours numerically and as close as possible to how humans perceive colours visually.

2.3.3 CIE Colour Space

CIE is the abbreviated form of its French title "Commission Internationale de l'Eclairage" also known as the International Commission on Illumination in English. This organisation is focused on exchanging studies and researches between the member countries on an international scale in any topic related to science and lighting

The CIE was designed to classify colour to be closer to the Human Visual System (HVS). Considering the 3-cone system that the human eye is based on, CIE developed the tristimulus system after measuring the sensitivities of the cones in our eyes. This system of translating colours to numerical values has been defined as the CIE XYZ colour space. This colour space was derived from the studies conducted in the late 1920s by William David Wright and John Guild which was known as CIE 1931 RGB colour space and CIE 1931 XYZ colour space (Pointer 1981).

The X, Y and Z in the CIE XYZ colour space are the values represented for the various properties of the HVS.

X – a mixture of positive values

Y – a value set for luminance

Z – a value approximately equal to blue

The CIE XYZ is a device-independent colour space. Keeping the CIE XYZ as a base other colour spaces in the CIE space were developed which are also device-independent. These colour spaces include the widely used formats CIELab and CIELuv.

The goal to introduce the two new colour spaces CIELab and CIELuv in 1976, was to create a perceptually equal colour space. In other words, the Euclidean distance calculated between two different colours which were present in the CIELab or CIELuv colour spaces has a strong correlation with the HVS. The main difference between the two models is in the implementation of the model for chromatic adaptation. The CIELab model normalises its values by dividing the white point while the CIELuv model subtracts the white point. A point to consider for both CIELab and CIELuv is that both the models are non-linear colour spaces (Pointer 1981). Conversions between RGB and CIELab/CIELuv models are not available as RGB is a device-dependent colour space while CIE is device-independent.

The figure 20 is the CIE 1931 Chromaticity diagram. The outer curved boundary is the monochromatic locus and the values shown are the wavelengths displayed in nanometres. The colours shown would differ based on the medium of display used as the colours are specified in the sRGB format and some colours lying outside the range of sRGB would be displayed differently depending on the display.

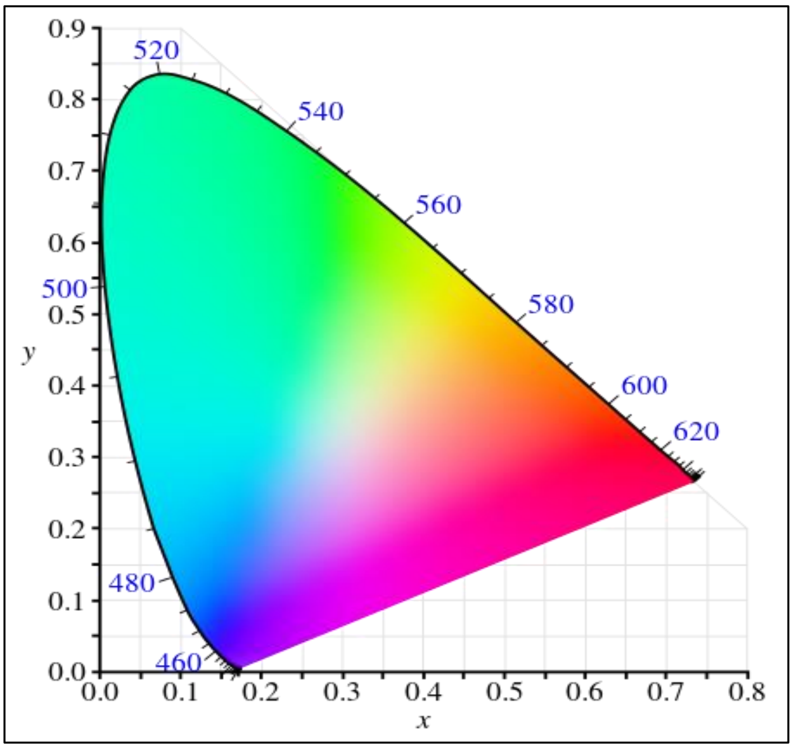


Figure 20: CIE 1931 colour space chromaticity diagram

Even if the CIE model was designed to be as close to human perception as possible, it must also be noted that every human perceives colour differently depending on the nature of the cones in their eyes. Keeping this fact in mind, the CIE has defined a colour mapping function called the “Standard Observer” to represent an average human being. This was set to an average person’s response to colour within a 2° arc inside the area of the eye where the cones are located, known as the fovea. This function is also known as the CIE 1931 2° Standard Observer for this reason. A disadvantage of this model which is similar to the RGB model is the non-linearity of the colours making it difficult to find the similarity between colours. Another drawback is the high computation costs, which does not make sense for applications with small data sets. An updated version of CIE XYZ, the CIE LAB model removes the drawback of non-linearity but the high computation costs remain.

2.3.4 Munsell Colour Space

Munsell Colour Space is another colour space similar to the HSV/HSL colour model as it is also based on Hue, Saturation/Chroma and Lightness properties of colour (A.H. Munsell 1907). It was developed by Professor Albert H. Munsell in the early 20th century. This colour space has been adopted as the official colour system for soil research by the United States Department of Agriculture since the 1930s (Munsell Color System and Color Matching from Munsell Color Company 2011a).

The Munsell Colour theory is based on the Munsell Colour Tree, which is a three-dimensional irregular colour solid based on the three properties of colour – Hue, Chroma and Value. In this model –

Hue – is measured in degrees along the horizontal axis

Chroma – is measured from the centre to the outward section radially starting from the neutral grey colour.

Value – is measured vertically at the centre of the model starting from 0 (Black) at the bottom to 10 (White) at the top.

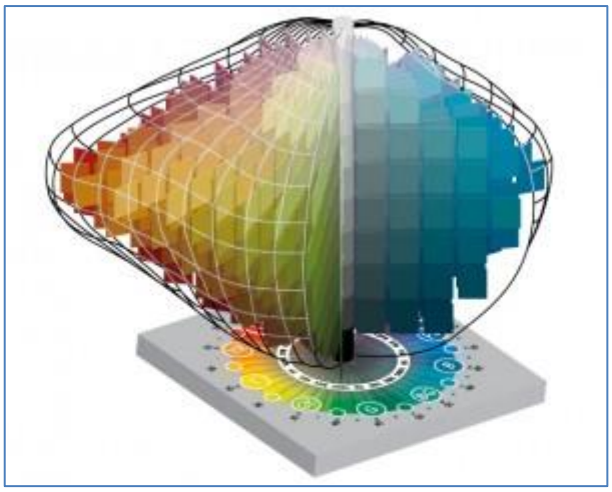


Figure 21: Munsell Colour Solid (Munsell Color System and Color Matching from Munsell Color Company 2011c)

This arrangement is also known as the Munsell Colour Space. And all the colour lies within a particular section of the Munsell colour space which is known as the Munsell Colour Solid. The irregular shape of this model is due to Munsell determining the space between the colours by measuring the human visual responses. (A.H. Munsell 1907) (Munsell Color System and Color Matching from Munsell Color Company 2011c)

2.3.4.1 Munsell Hue

As defined earlier, Hue is the sensation to understand the difference between shades of colours and perceive how similar or different one colour is to another. But, Munsell decided to divide the Hue circle based on the colours and define regions based on the colours. Munsell named the colours red, yellow, green, blue, and purple as “principal hues” and positioned them at equal angles around a circle. He also added five subsections between the principal hues namely, yellow-red, blue-green, purple-blue, and red-purple. He simplified these 10 hues as R, YR, Y, GY, G, BG, B, PB, P and RP which are the initials of the actual hues.

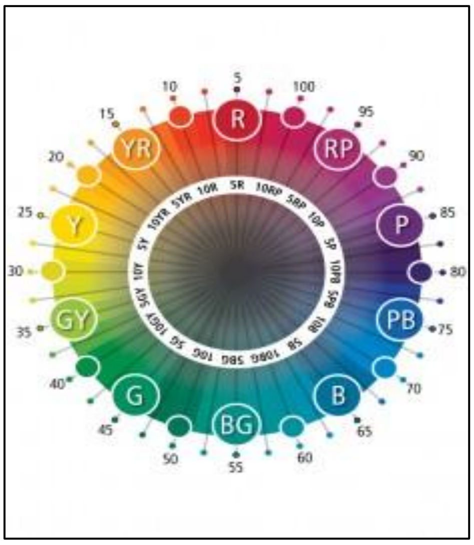


Figure 22: Depiction of Hue placement in a circle by Munsell (Munsell Color System and Color Matching from Munsell Color Company 2011d)

Munsell divided the Hue circle into 100 steps starting from Red, which was placed at zero. But the hue ranges are identified by the Hue sector and the steps are on the scale of 10 and zero is not used. The figure below shows the categorisation of the Munsell Hue values at roughly the mid-point at value 6 and chroma 6.

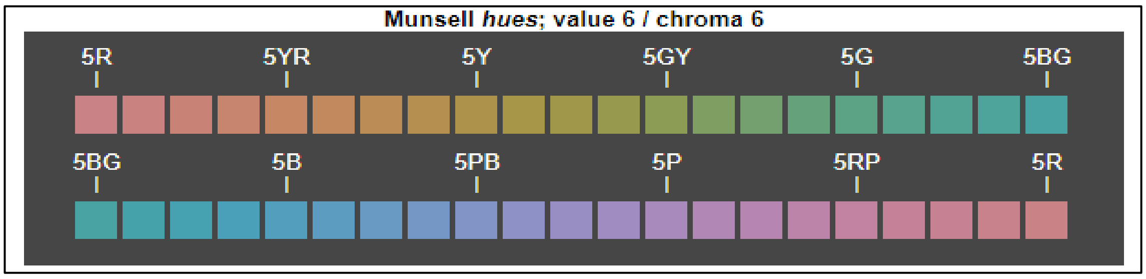


Figure 23: Example of a section of Munsell Hue values (Wikipedia 2021a)

2.3.4.2 Munsell Chroma

Chroma as defined previously is the sensation of the colourfulness of an object when compared to a white background (Munsell Color System and Color Matching from Munsell Color Company 2011b). In other words, it is the strength of the hue. Colours with low chroma are closer to grey and are sometimes known as “weak” colours and colours having high chroma values are sometimes referred to as being highly saturated colours or being strong or vivid colours. There is usually no upper limit for chroma values and different areas in the colour space have different values. Thus, the Munsell colour solid being irregular in shape.

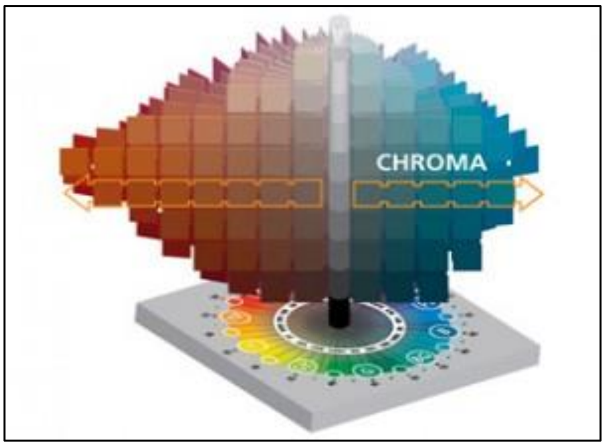


Figure 24: Chroma in the colour model as designed by Munsell (Munsell Color System and Color Matching from Munsell Color Company 2011b)

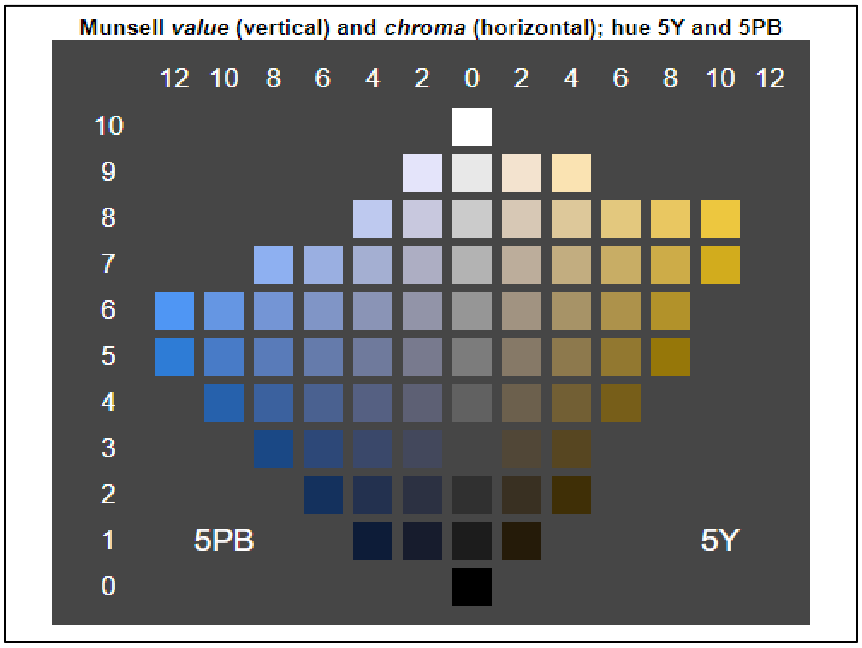


Figure 25: Example of Munsell Chroma and Munsell Value (Wikipedia 2021a)

2.3.4.3 Munsell Value

Munsell values are the indication of how light a colour is in the Munsell colour solid. The values range from 0 (pure black) to 10 (pure white). The grey colours in this range including white and black are known as “neutral colours” as they have no hue.

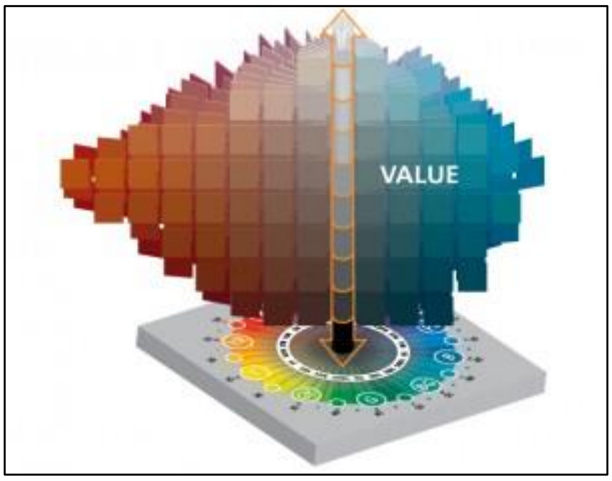


Figure 26: Munsell Value (Munsell Color System and Color Matching from Munsell Color Company 2011e)

A good representation to understand the Munsell Hue, Chroma and Value in the Munsell Colour Solid is the image below.

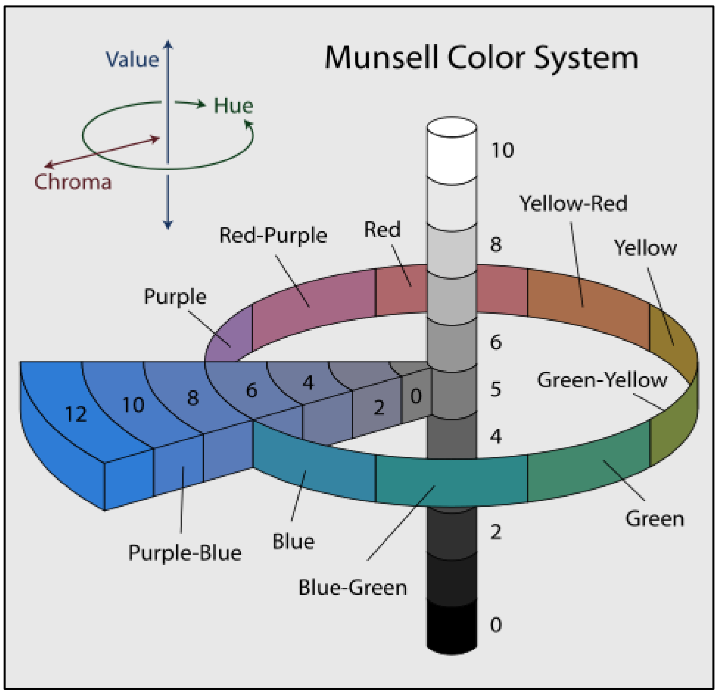


Figure 27: Munsell Colour System (Wikipedia 2021a)

2.3.5 Pantone Matching System

The Pantone Matching System (PMS) was developed in the 1950s by Pantone LLC, a company headquartered in New Jersey, US (History of Pantone Inc. – FundingUniverse 2020). This system was developed to standardise colours to make it easier for production. One of the major problems designers and brand owners faced was that the manufactured product might not have the same colour that they used to design their product. The PMS solved this problem by defining a code for each colour and made it a standard across industries. Because of this system, the digital version and the final manufactured product has the same colour as intended by the designer.

Pantone manufactures the Pantone Guide which consists of various colours and their corresponding codes printed on thin cardboard sheets in a "fan deck" manner so that users can colour match their products. JewelCandle uses the Pantone naming scheme for the colours used in the designing of its products to maintain the uniformity of colours when there would be a need to possibly reuse the same colour for another product. This is an important fact to be considered since the Pantone codes will need to be converted to an appropriate format for further analysis. This is discussed further in chapter 3.

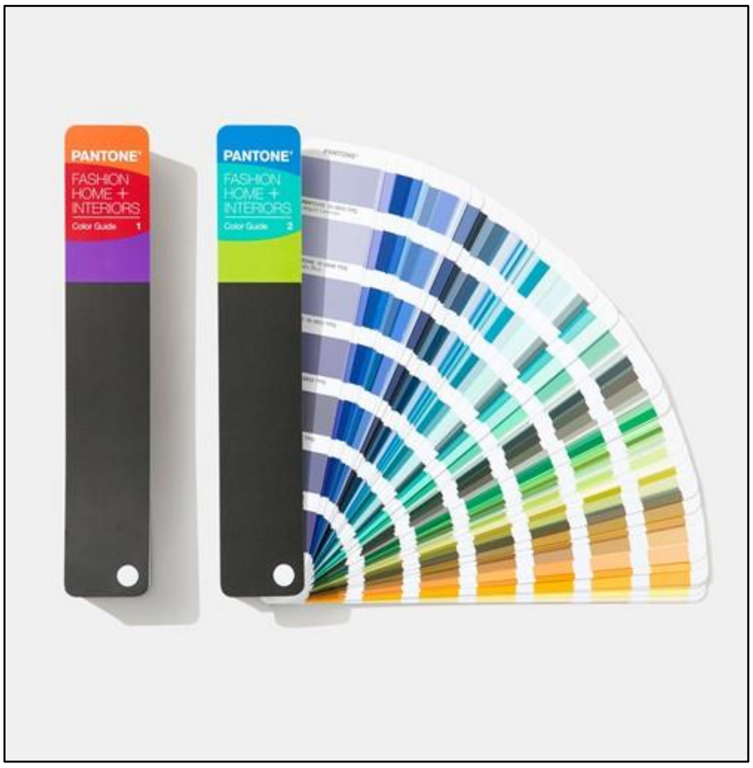


Figure 28: Example Pantone Guide (Pantone 2020)

Pantone has asserted that the colour numbers and values are their intellectual property and free use of these resources are not allowed. This makes it very expensive and difficult to use it in an open-source solution for any purpose.

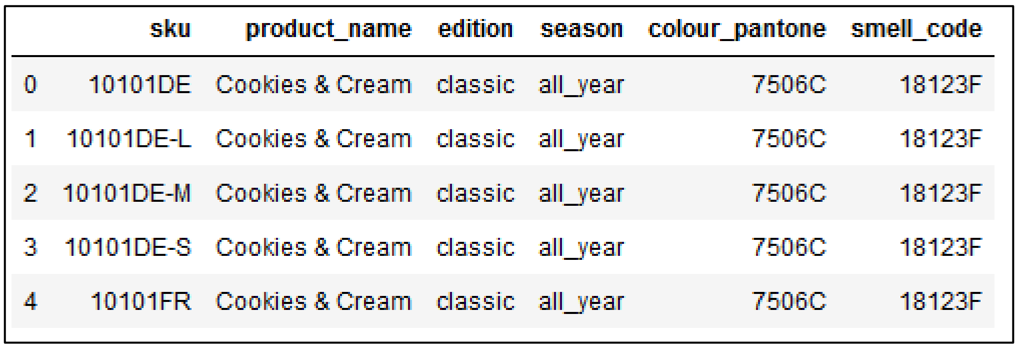
In this chapter, we have explored the human visual system and the perception of colours by humans. Following which we tried to understand the potential colour spaces that could be used. Starting from the RGB colour model where it was seen that it is not very similar to human perception and is not the most efficient model for our use case since it is not linear in nature and would need further processing to make it linear. Although the CIE model would be the better option to consider if only human perception had to be considered, the computation costs relative to the dataset must also be considered. The Munsell model was also developed to be similar to human perception but its focus has been on soil colours and there are missing sections within the model which was covered in the HSL model. After understanding the various models, we learn that out of various models the HSV/HSL model seems to be ideal for our use case since it is linear making it easier to calculate the differences and it is fairly similar to how humans perceive colours. In this case, the HSV model lies in the sweet spot of low computational costs and decent colour perception. Keeping the model in mind the next step would be to understand the available data and make necessary transformations as and where required to form a dataset that could be easily used to build the models. The next chapter describes the data understanding and preparation process to convert the Pantone colour codes to their respective HSL values to continue with our analysis.

3 Data Understanding and preparation

After determining the colour model that would be the most suitable for our use case, the next step is to understand the available data related to the products stored in the company's product database. This chapter focuses on the different attributes available for the products and how they would be manipulated to form the ideal structure that would be required to build the different models. The focus is particularly on the only colour related field available which is the Pantone code of the colour used for the product. We start by examining all the available attributes and shortlisting only the ones that would be needed and then further perform the required manipulation.

3.1 Data available from the source

The product information is stored in a database with many attributes to a product. Some of these are "*sku*" which is the product code, "*product_name*" which is the name of the product, "*edition*" which stores values such as "*classic*" and "*exclusive*", amongst others, "*glass_type*" stores the value stating the size of the glass jar of the candle, colour related attributes such as "*colour_name*" and "*colour_pantone*" that stores the general name of the colour and the Pantone code of the product respectively. Apart from these attributes, there are smell related attributes such as the "*smell_code*", "*smell_name*", etc. and the surprise gift related attributes such as "*jewellery_type*" (ring, necklace, bracelet, etc), "*jewellery_material*" (gold or silver). But for this proof of concept, a subset of the complete attributes has been extracted based on the initial requirements. These particular attributes, which include the "*sku*", "*product_name*", "*edition*", "season", "*colour_pantone*" and "*smell_code*", were decided on based on its uniqueness to the product and its relation to the analysis conducted in this project. A sample dataset of the extracted features looks like below.



	sku	product_name	edition	season	colour_pantone	smell_code
0	10101DE	Cookies & Cream	classic	all_year	7506C	18123F
1	10101DE-L	Cookies & Cream	classic	all_year	7506C	18123F
2	10101DE-M	Cookies & Cream	classic	all_year	7506C	18123F
3	10101DE-S	Cookies & Cream	classic	all_year	7506C	18123F
4	10101FR	Cookies & Cream	classic	all_year	7506C	18123F

Figure 29: Sample of the extracted dataset from the products database being used.

The description of each field is as below.

Attribute Name	Description
sku	Unique product code for each product
product_name	Name of the product
edition	Edition of the product. These are classic, exclusive, Swarovski etc.
season	The season when the product is available in the shop. They are all_year, summer and winter.
colour_pantone	The Pantone code for the colour of each product.
smell_code	The code of the perfume used in the candle.

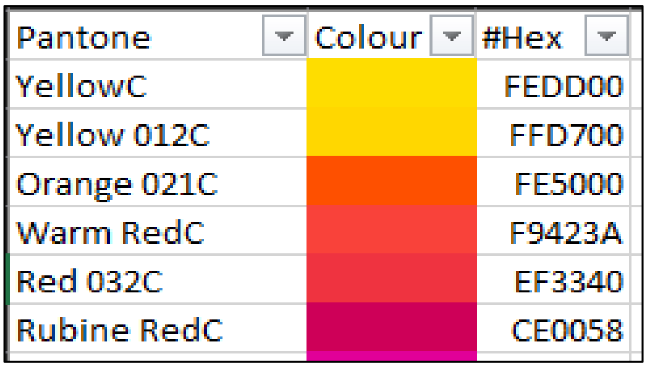
Table 1: Column description of the extracted dataset depicted in Figure 29

The extracted dataset will be further reduced to use just the "*product_name*" and "*colour_pantone*". As seen in the sample dataset, the Pantone code for the colour is stored in the backend database. And from the findings in chapter 3.2.6, these codes cannot be used freely. There is no logical conversion of Pantone codes to any other colour format. The only way to do this is to map each Pantone code to a Hex value and then use that Hex value to convert to any other colour space as required. The further data preparation is explained in the next chapter.

3.2 Data preparation

The conversion of Pantone codes (PMS) to hex value would ideally have to be done by manually comparing each Pantone using the tool available at pantone.com (pantone.com). But this was simplified with the help of a readily available list of the hex codes for most of the Pantone codes produced (PMS to Hex Color Chart | Pantone Color to Hex Conversion 2020) and this was enough for this use case. Using this source would not be needed in the future as a change in the data structure has been introduced in the new product management system where the Hex values would also be stored.

The extracted converted data looks like below. The "Colour" column would not be required and is in the file only for personal reference. The column is dropped while merging with the other dataset.



Pantone	Colour	#Hex
YellowC	🟨	FEDD00
Yellow 012C	🟨	FFD700
Orange 021C	🟧	FE5000
Warm RedC	🟥	F9423A
Red 032C	🟥	EF3340
Rubine RedC	🟥	CE0058

Figure 30: PMS to Hex converted values obtained from (PMS to Hex Color Chart | Pantone Color to Hex Conversion 2020)

Now, as per the understanding from chapter 3.2.3 where the HSL/HSV colour space was described and for further processing, PMS values need to be converted to HSL values. The conversion takes place as per the following process.

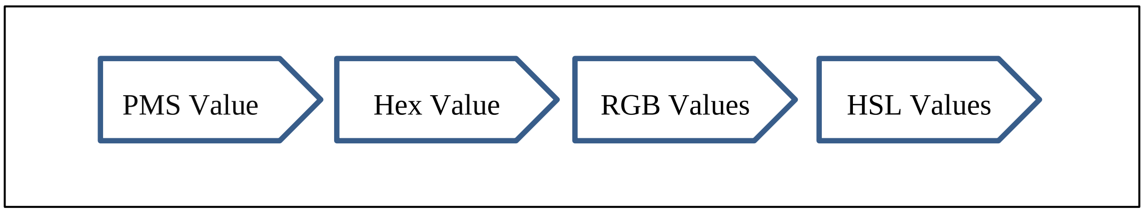


Figure 31: PMS to HSL Values conversion process

The first part of converting PMS to Hex has been handled and the next step is to convert the Hex values to RGB. All the codes have been written in the python scripting language. Hex values are the hexadecimal representations of RGB values. Hex values are represented with a “#” prefix to 6 characters or numerical digits. 2 characters represent the values for each of Red, Blue and Green values. For example, #FFFFFF is the hex value for the colour white and the RGB values for the same are R:255 G:255 B:255. To achieve this the “#” character from the Hex values have to be first removed. Then the remaining 6 characters have to be split into pairs of 2 characters. These pairs are then

```python
# Function to convert hex values to RGB values

def hex_to_rgb(hex_value):
    h = str(hex_value).lstrip('#')
    return tuple(int(h[i:i + 2], 16) / 255.0 for i in (0, 2, 4))
```

Code Snippet 1: Python function to convert Hex values to RGB Values converted from their hexadecimal format to decimal format. This conversion is done easily with the below function in python.

The division by 255 in the function is to keep the RGB values in the range of 0 and 1. This is necessary for the next conversion to HSL values. Now, the RGB values are converted to HSL values using the logic explained in chapter 3.2.3.1. This was achieved using the python function below.

```python
# Function to convert RGB values to HSL values

def rgb_to_hsl(r, g, b):
    r = float(r)  # Value of Red attribute
    g = float(g)  # Value of Green attribute
    b = float(b)  # Value of Blue  attribute
    high = max(r, g, b)
    low = min(r, g, b)
    h, s, l = ((high + low) / 2,)*3  # Storing the values of Hue (h), Satura-
    tion(s) and lightness(l)

        if high == low:
            h = 0.0
            s = 0.0
        else:
            l = (high + low) / 2
            d = high - low
            s = d / (2 - high - low) if l > 0.5 else d / (high + low)
            h = {
                r: (g - b) / d + (6 if g < b else 0),
                g: (b - r) / d + 2,
                b: (r - g) / d + 4,
            }[high]
            h /= 6

        return h, s, l
```

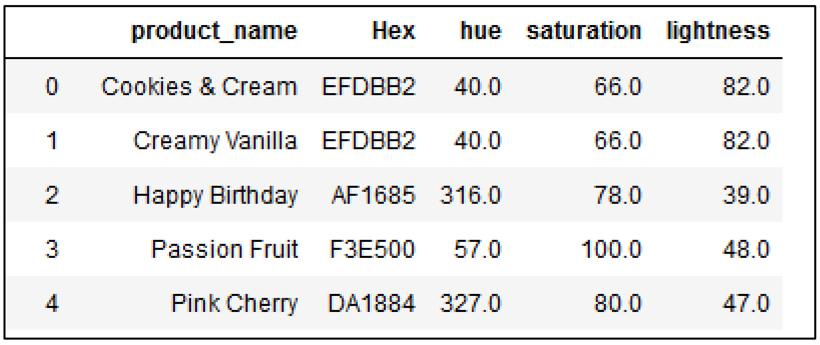
Code Snippet 2: Python function to convert RGB to HSL values

Finally using the 2 functions and the PMS to Hex mapping file, the HSL values are added to the mapping file using the code below. Please note the further calculations included in the Hue, Saturation and Lightness values. Since the hue is represented as a circle, the resultant value needs to be multiplied by 360 to convert the values to degrees. And the saturation and lightness values need to be multiplied by 100 since they are percentages. The logic of the code is such that for every value of Hex present in the dataset, the hue, saturation and lightness are calculated and stored in their respective lists. These lists are then appended to the ”*pantone_hex*” data frame.

```python
# Calculate hue for each colour

# The use of pantone_hex dataset would not be needed as the hex val-
   ues would be used directly from
# the main dataset once available


hue=[]
saturation=[]
lightness=[]

for i in range(len(pantone_hex.Hex)):
    hue.append(
        round(
            rgb_to_hsl(
                hex_to_rgb(pantone_hex.Hex.astype(str)[i])[0],
                hex_to_rgb(pantone_hex.Hex.astype(str)[i])[1],
                hex_to_rgb(pantone_hex.Hex.astype(str)[i])[2])[0]*360)
    )

for i in range(len(pantone_hex.Hex)):
    saturation.append(
        round(
            rgb_to_hsl(
                hex_to_rgb(pantone_hex.Hex.astype(str)[i])[0],
                hex_to_rgb(pantone_hex.Hex.astype(str)[i])[1],
                hex_to_rgb(pantone_hex.Hex.astype(str)[i])[2])[1]*100)
    )

for i in range(len(pantone_hex.Hex)):
    lightness.append(
        round(
            rgb_to_hsl(
                hex_to_rgb(pantone_hex.Hex.astype(str)[i])[0],
                hex_to_rgb(pantone_hex.Hex.astype(str)[i])[1],
                hex_to_rgb(pantone_hex.Hex.astype(str)[i])[2])[2]*100)
    )

pantone_hex['hue']=hue
pantone_hex['saturation']=saturation
pantone_hex['lightness']=lightness
```

Code Snippet 3: Python code to obtain the HSL values from PMS

After obtaining the HSL values, the final dataset looks as in figure 32. The "*sku*", "*edition*", "*season*", "*smell_code*" columns were not needed for further processing and the "*colour_pantone*" attribute was converted to "*Hex*", "*hue*", "*saturation*" and "*lightness*" attribute which would be used for the categorisation. The "*Hex"* attribute although not necessary, is kept only for reference. This the final dataset that would be used to test different algorithms and functions.



	product_name	Hex	hue	saturation	lightness
0	Cookies & Cream	EFDBB2	40.0	66.0	82.0
1	Creamy Vanilla	EFDBB2	40.0	66.0	82.0
2	Happy Birthday	AF1685	316.0	78.0	39.0
3	Passion Fruit	F3E500	57.0	100.0	48.0
4	Pink Cherry	DA1884	327.0	80.0	47.0

Figure 32: Final dataset generated after the conversion of Pantone codes to Hex and HSL values

Although this dataset will be used for almost all the algorithms that have been experimented with, there is a need to further process this dataset for the use of a custom function that is described in chapter 5.1. The Hue, Saturation and Lightness values will be further split into categories based on visual similarities. This split would be generated based on the visual and angular differences noticed with the help of figure 17.

This chapter focused on the cleaning and manipulation of the available data from the backend products database. Since the database only stored the Pantone codes of the colours it was necessary to convert it to the HSL format, and this was not a direct conversion due to the limitations of the Pantone colour format w. Having finalised the structure of the dataset that would be used, the next step is to use this dataset with multiple algorithms and functions. The next chapter and its sub-chapters describe the various algorithms and functions that were experimented with to understand which algorithm would work best for this particular proof of concept.

4 Experimentation and modelling

As understood in the introduction of this thesis the objective of this chapter is to explore possible solutions to cluster the products based on their colours. This process was started by first finalising the colour model that we would be working with. We then explored the available dataset from the company’s product database and extracted the required features and further manipulated them to suit our algorithms. Having prepared the dataset to work with, we now proceed with the exploration of different mathematical models and unsupervised machine learning models. A computer algorithm determines the clusters by evaluating “distances” between the different data points in the dataset. These distances are mathematically calculated using various distance formulas, for example, the Euclidean distance formula, to calculate the distances between attributes, which in our case are the hue, saturation and lightness. Apart from readily available algorithms to work with, a custom function was designed which makes use of visual differences seen by a regular human eye and by splitting the attribute of hue, saturation and lightness manually based on the HSL colour wheel as seen in figure 33 for clustering. All these algorithms use the hue, saturation and lightness values that were generated from the to cluster all the different colours. These models are being explored as potential solutions to clustering the different colours of the products in our portfolio. All these algorithms and functions are described in more details below.

4.1 Manual categorisation based on visual differences

As described briefly in the previous section, this function was designed by categorising the Hue, Saturation and Lightness values based on visual differences. As mentioned in chapter 3.2, the dataset that was prepared would need further processing just for this custom function. While the other algorithms are either modified versions of mathematical formulas or machine learning algorithms that are readily available to implement, this processing takes place in the form of splitting the generated hue, saturation and lightness values of each colour based on the HSL colour wheel depicted in figure 33.

4.1.1 Data preparation

As seen in the calculation in chapter 3.2, Hue ranges from 0 – 360 as it represents a circle, while Saturation and Lightness are in the range of 0 – 100 since they are percentages. The visual differences for Hue values were based on figure 33.

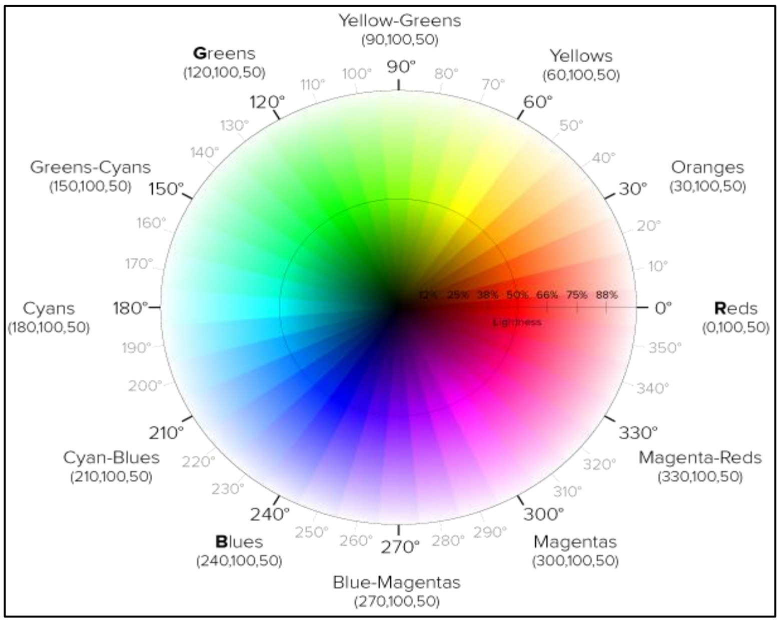


Figure 33: HSL colour wheel depicting the splits of colours by the degrees of the circle (Codrops 2016)

The Hue values were broadly categorised for every 30 degrees considering figure 33 as a reference. The implementation of this logic has been shown in code snippet 4. The major groups are then sub-categorised into hue sub-groups by splitting each

```
# Define the major colour groups for each Hue

def hue_group(i):
    if   pantone_hex.hue[i] in range(330,361) : return 'Magenta-Reds'
    elif pantone_hex.hue[i] in range(300,330) : return 'Magentas'
    elif pantone_hex.hue[i] in range(270,300) : return 'Blue-Magentas'
    elif pantone_hex.hue[i] in range(240,270) : return 'Blues'
    elif pantone_hex.hue[i] in range(210,240) : return 'Cyan-Blues'
    elif pantone_hex.hue[i] in range(180,210) : return 'Cyans'
    elif pantone_hex.hue[i] in range(150,180) : return 'Greens-Cyans'
    elif pantone_hex.hue[i] in range(120,150) : return 'Greens'
    elif pantone_hex.hue[i] in range(90 ,120) : return 'Yellow-Greens'
    elif pantone_hex.hue[i] in range(60 , 90) : return 'Yellows'
    elif pantone_hex.hue[i] in range(30 , 60) : return 'Oranges'
    else: return 'Reds'
  # Create list of colour groups
hg=[]
for i in range(len(pantone_hex.hue)):
    hg.append(hue_group(i))

# Add the colour groups to dataframe
pantone_hex['hue_group']=hg
```

Code Snippet 4: Code snippet to manually assign groups for Hues of the 30 degrees group into smaller groups of 10 degrees each. Code snippet 5 shows how this logic was achieved in the form of a python function.

```
# Splitting each Hue group into sub groups ranging from 0-360

def hue_sub_group(i):
    if pantone_hex.hue_group[i]=='Magenta-Reds' :
        if pantone_hex.hue[i] in range(330,340): return('MR1')
        elif pantone_hex.hue[i] in range(340,350): return('MR2')
        else: return('MR3')
    elif pantone_hex.hue_group[i]=='Magentas' :
        if pantone_hex.hue[i] in range(300,310): return('M1')
        elif pantone_hex.hue[i] in range(310,320): return('M2')
        else: return('M3')
.
.  # Code snippet shortened for the purpose of example
.
.
    else:
        if pantone_hex.hue[i] in range(10): return('R1')
        elif pantone_hex.hue[i] in range(10,20): return('R2')
        else: return('R3')


# Create list of hue sub groups
hsg=[]
for i in range(len(pantone_hex.hue)):
    hsg.append(hue_sub_group(i))

# Add the colour groups to dataframe
pantone_hex['hue_sub_group']=hsg
```

Code Snippet 5: Creating sub-groups for each generated Hue group

Next, the groups for Lightness and Saturation values are generated. Both of these attributes are values percentage values ranging from 0-100%. Due to this reason, they have been split into ranges of 20 units. Code snippets 6 and 7 show the logic of this function.

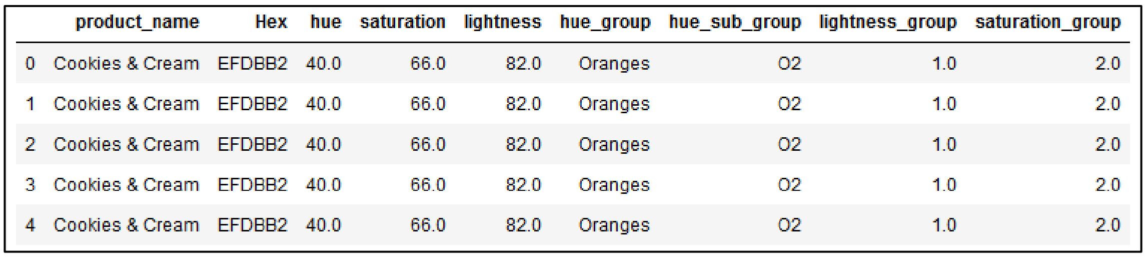
```
# Define the lightness groups for each Hue
# checking the index of descriptive string values did not give appropri-
 ate results, so switching to sequential tags
  def lightness_group(i):
    if   pantone_hex.lightness[i]>=80 : return int(1)
    elif pantone_hex.lightness[i] in range(60 , 80) : return int(2)
    elif pantone_hex.lightness[i] in range(40 , 60) : return int(3)
    elif pantone_hex.lightness[i] in range(20 , 40) : return int(4)
    else: return int(5)
  # Create list of colour groups
lg=[]
for i in range(len(pantone_hex.hue)):
    lg.append(lightness_group(i))
  # Add the colour groups to dataframe
pantone_hex['lightness_group']=lg
```

Code Snippet 6: Function to create groups for Lightness attribute of colour

```python
# Define the saturation groups for each Hue
# checking the index of descriptive string values did not give appropriate results, so switching to sequential tags

def saturation_group(i):
    if   pantone_hex.saturation[i]>=80 : return int(1)
    elif pantone_hex.saturation[i] in range(60 , 80) : return int(2)
    elif pantone_hex.saturation[i] in range(40 , 60) : return int(3)
    elif pantone_hex.saturation[i] in range(20 , 40) : return int(4)
    else: return int(5)

# Create list of colour groups
sg=[]
for i in range(len(pantone_hex.hue)):
    sg.append(saturation_group(i))

# Add the colour groups to dataframe
pantone_hex['saturation_group']=sg
```

Code Snippet 5: Function to create groups for Saturation attribute of colour

After these additions to the dataset, it forms the structure as shown in figure 34 below. As mentioned previously, this dataset is being used only for this particular function and all the remaining algorithms and functions use the final dataset structure created in chapter 3.2. This structure shown in figure 34 is the final result of the “manual clustering” that was performed based on the visual differences. These are not programmatic and have been generated based on personal understanding and can be subjective.



	product_name	Hex	hue	saturation	lightness	hue_group	hue_sub_group	lightness_group	saturation_group
0	Cookies & Cream	EFDBB2	40.0	66.0	82.0	Oranges	O2	1.0	2.0
1	Cookies & Cream	EFDBB2	40.0	66.0	82.0	Oranges	O2	1.0	2.0
2	Cookies & Cream	EFDBB2	40.0	66.0	82.0	Oranges	O2	1.0	2.0
3	Cookies & Cream	EFDBB2	40.0	66.0	82.0	Oranges	O2	1.0	2.0
4	Cookies & Cream	EFDBB2	40.0	66.0	82.0	Oranges	O2	1.0	2.0

Figure 34: Snapshot of the final dataset created for manual categorisation of colours

The next chapter describes the logic and working of this custom function.

4.1.2 Working logic of the function

The basic idea behind this function is to take the input “product” and compare the values of the newly created groups and sub-groups of the colour attributes, i.e., “*hue_group*”, “*hue_sub_group*”, “*saturation_group*” and “*lightness_group*”, with the same attributes of every other product in the dataset. The comparison logic is split into two parts. The first part of the function retrieves a list of similar products based on certain parameters for the colour attributes of hue, saturation and lightness. The second section of the function is used to dynamically determine the number of returned products as well as set the values of the attributes depending on the results of the previous run of the function.

Part 1 –

```python
# Get list of similar looking candles based on colour parameters and main candle

def col_rec(candle, col_list, h1, h2, l1, l2, s1, s2):
    R = prods_new[col_list][(prods_new.product_name!=candle)
                            &
                        (prods_new.hue.isin(list(range(int(prods_new.hue[prods_new.p
   roduct_name==candle].unique())-h1,
                                                       int(prods_new.hue[prods_new.p
   roduct_name==candle].unique())+h2)
                                                ))
                        )&
                        (
                            (prods_new.lightness_group.isin(list(range
                                                (int(prods_new.light-
   ness_group[prods_new.product_name==candle].unique())-l1,
                                                 int(prods_new.light-
   ness_group[prods_new.product_name==candle].unique())+l2)
                            )))&
                            (prods_new.saturation_group.isin(list(range
                                                (int(prods_new.satura-
   tion_group[prods_new.product_name==candle].unique())-s1,
                                                 int(prods_new.satura-
   tion_group[prods_new.product_name==candle].unique())+s2)
                            ))))
                            ].drop_duplicates(subset ="product_name").sort_val-
   ues(by=['hue']).reset_index(drop=True)
    return(R)
```

Code Snippet 6: Function to search for similar products based on attribute ranges.

This is a function to check the complete list of products whose attributes are within a range of the input product. The function initially used just the main Hue group and present all the products within the same group. But this was inefficient and had to be modified to check each attribute and the sub-attributes.

As seen in code snippet 8, this function takes in the input candle name, the list of columns to be extracted and the upper and lower limit for the values of Hue, Saturation and Lightness values that have to be set in the ranges. These variables are included in the filter condition for the search and only the products that match all the criteria are returned. The input parameters h1, h2, s1, s2, l1 and l2 are set by the function defined in Part 2.

Part 2 –

This part consists of a function that dynamically sets the range of values for the colour attributes. This function also checks the number of products that were produced in the search and dynamically increases the range of values if the number of products returned does is less than the number of “recommendations” needed. The number of recommendations is set as an input parameter for the function.

As seen in code snippet 9, there are initial values set for each of the attribute ranges. Starting from a difference of 1 unit and then increasing it as per the number of products retrieved. The first check is done by comparing the number of products retrieved from the search function based on the base ranges. If the number of products retrieved is less than the required number, then the value of Hue is increased by 1. Then the output is checked again for the number of products retrieved. Supposing a situation where the function is not able to reach the required number by the time the Hue value has been increased by 5, the values of Saturation and Lightness are increased and then the results are checked again. The value of Hue is still being increased while this loop executes. The check for the required number of products is done during every loop and if the function is still not able to find the required number then the values of Saturation and Lightness have increased again when the value of Hue has been increased by more than 10 units. Some products have very unique colours and are very few in numbers. Due to this fact, the function was not modified to add more checks.

```python
## Dynamically increase parameter values based on the number of recommended can-
   dles retreived
## Logic needs tweaking.

def M_1(candle, col_list, len_recc=5):
    #len_recc=5
    h1=1
    h2=2
    l1=1
    l2=2
    s1=1
    s2=2
    R1 = col_rec(candle, col_list, h1=h1, h2=h2, l1=l1, l2=l2, s1=s1, s2=s2)
    while len(R1)<len_recc:
        h1+=1
        h2+=1

### Add logic to increase range of lightness and saturation based if hue range in-
   creases after a set threshold
        if h1>5 & h2>5:
            l1=2
            l2=3
            s1=2
            s2=3
        elif h1>10 & h2>10:
            l1=3
            l2=4
            s1=3
            s2=4

        t1 = col_rec(can-
   dle, col_list, h1=h1, h2=h2, l1=l1, l2=l2, s1=s1, s2=s2).head(len_recc-len(R1))
        R1=R1.append(t1[~t1.product_name.isin(list(R1.product_name.unique()))]).re-
   set_index(drop=True)

    return(R1)
```

Code Snippet 7: Python function to dynamically set the input parameters required for the product search function

The next chapter describes the function designed to calculate the Euclidean distance between two colours.

4.2 Categorisation based on the Euclidean distance between colours

This function was based on the concept of calculating the Euclidean distance between points in a three-dimensional space, where Hue, Saturation and Lightness are the three dimensions in the HSL colour space. This function does not create clusters for products but rather provides a set number of recommendations directly instead. The idea behind this function is to calculate the Euclidean distance between the input product and all the other products available in our dataset and retrieve the top N products having the least distance with the input product.

The mathematical concept of Euclidean distance is defined as the shortest distance or the length of a line segment between two points in a given space. The formula for the calculation of the distance between two points *A* (x_1, y_1) and *B* (x_2, y_2) in a two-dimensional space in the cartesian coordinate system is an interpretation of the Pythagorean theorem. The formula is as below.

$$d(A, B)=\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

This formula can be further modified to accommodate points in n-dimensional space. E.g., if the two points *A* ($x_1, x_2, x_3, \ldots, x_n$) *and B* ($y_1, y_2, y_3, \ldots, y_n$) then the distance between the two points would be calculated by the formula below.

$$d(A, B)=\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}$$

In our particular case, we have three dimensions, namely, Hue, Saturation and Lightness. Therefore, the Euclidean distance formula would be modified for a three-dimensional which would simply be as below.

$$d(A, B)=\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}$$

Now, the next part that has to be considered is that the value of Hue is in degrees. Since degrees cannot be considered to calculate distances, it is necessary to convert them into radians.

A radian is defined as "*the measure of an angle that, if placed at the centre of a circle, subtends (intersects) an arc of length equal to the radius of the circle.*" (Fred Safier 2003)

Considering this conversion and the Euclidean distance formula for points in three-dimensions, the python function was designed as below.

```python
def colour_dist(x, y):  # x and y are the input products
    import numpy as np
    dist = np.sqrt((x['saturation']*np.cos(x['hue']) - y['satura-
tion']*np.cos(y['hue']))**2 +\
    (x['saturation']*np.sin(x['hue']) - y['satura-
tion']*np.sin(y['hue']))**2 +\
    (x['lightness'] - y['lightness'])**2)
    return dist
```

Code Snippet 8: Function to calculate the distance between two points in the HSL colour

After defining the function to calculate the distances, the next part is to define the recommender function. This part will be split into three sections to explain the code.

Section 1 –

```python
### Method 2 - Euclidean distance between 2 colours based on Hue, Saturation and Lightness values of the colour

def euc_dist_recommend(candle, col_list):
    col_list_tmp = col_list.copy()  # Create a copy of the list of columns

    # Main Candle

    x=prods_new[col_list_tmp][prods_new.product_name==candle].\
                drop_duplicates(subset ="product_name").sort_values\
                (by=['hue', 'saturation', 'lightness']).reset_index(drop=True)
```

Code Snippet 9: Function definition for Euclidean distance calculation function

This section starts by defining the function. The inputs for this function are the name of the candle and the names of the required columns. Then a copy of the column list is created where an additional column is added in the latter part of the function. Next, a data frame is extracted from the entire list of products based on the input candle and the list of columns. It is made sure that no duplicates are present in this dataset.

Section 2 –

```python
#  get candles within set hue range for candles recommendation
    test_df = prods_new[col_list_tmp]\
                [(prods_new.product_name!=candle)&
                 (prods_new.hue.isin(
                     list(range(int(prods_new.hue[prods_new.product_name==candle].unique())-25,
                                int(prods_new.hue[prods_new.product_name==candle].unique())+26
                                )
                          ) # Hue range has been preset so that the colour difference does not vary too much
                 ))&
                 (prods_new.hue.notna())
                ].drop_duplicates(subset ="product_name").sort_values(\
        by=['hue', 'saturation', 'lightness']).reset_index(drop=True)
```

Code Snippet 10: Code snippet with function to extract data within set range

This section extracts a list of products that are in the range of ±25 units of Hue and stores it in a temporary data frame. This restriction of hue range was set to prevent the colour from changing too much from the expected range.

Section 3 –

```python
    # Get list of recommended candles based on euclidean distance of colours

    rows_list = []   # Create empty list to add rows to later convert to pandas dataframe
    for p in test_df.product_name:
        dict1 = {}   # Create empty dictionary
# Generate dictionary of products along with the calculated distance
        dict1.update({'product_name':list(test_df.product_name[test_df.product_name==p])[0],
         'Hex':list(test_df.Hex[test_df.product_name==p])[0],
         'hue':list(test_df.hue[test_df.product_name==p])[0],
         'lightness':list(test_df.lightness[test_df.product_name==p])[0],
         'saturation':list(test_df.saturation[test_df.product_name==p])[0],
         'dist':list(round(colour_dist(x,test_df[test_df.product_name==p].reset_index(drop=True)),2))[0]})
# Append the dictionary generated above into the empty list created previously
        rows_list.append(dict1)
    col_list_tmp.append('dist') # Add new column name 'dist'
    df = pd.DataFrame(rows_list, columns=col_list_tmp).sort_values(by='dist').reset_index(drop=True)

    return df.head(10) # Number of products set to 10. Can be modified to incoporate dynamic values.
```

Code Snippet 11: Section of function which calculates the distance and stores in a dataframe

In this section, a python dictionary is created containing the Hex, hue, lightness, and saturation columns along with an additional "*dist*" column where the values of the calculated distances between each product and the main product are stored. This is the part where the distance calculation function which was previously defined is used. One thing to notice here is that the number of "recommendations" are limited to 10. This can be changed easily by adding an input parameter for the same. This was not done since this was an experimental function.

One point that was noticed in this function was that even if the distances calculated were less between two products, there is a possibility that the Hue values were still far apart thus resulting in colours nowhere close to expectations. This was taken care of by adding a Hue range as seen in Section 2 of the code snippet. During experimentations, it was observed that this was an optimal range where enough products were "recommended" by the function. There are some cases where there are just 2-3 products in the range, but after inspecting manually there were no other products closer to the main product based on the colour.

An important fact to keep in mind with the functions described in chapters 4.1 and 4.2 is that these algorithms return the comparative results for the entire dataset. These functions also have to be executed every time a “recommendation” is required thus making it resource-heavy. The resources needed are computational power and electricity, and the more the dataset set and hits on the website increases the greater the consumption of these resources are. This is a major disadvantage of these algorithms. Moving away from designing a custom function and using a mathematical function to determine similar products, the next two chapters describe unsupervised machine learning algorithms that do this work.

4.3 Categorisation based on K-Means algorithm

K-means clustering algorithm is one of the simplest and most used clustering algorithms which has been used regularly for many problems similar to the one this thesis focuses on. This algorithm was developed by Sebestyen (1962) and MacQueen (1967) (Steinley 2006) independently to recognise ideal partitions in a group of data points. This development was motivated by the early researches done for maximising the homogeneity within a particular group.

The general idea behind the clustering is to partition a given set of data points into ‘K’ groups or clusters based on a particular attribute. This is a computationally intensive and complicated process that can be achieved easily in python using the KMeans package in the form of the Scikit Learn project (Fabian Pedregosa et al. 2011).

Figure 35 shows a general idea of how clustering looks like after using the K-means algorithm.

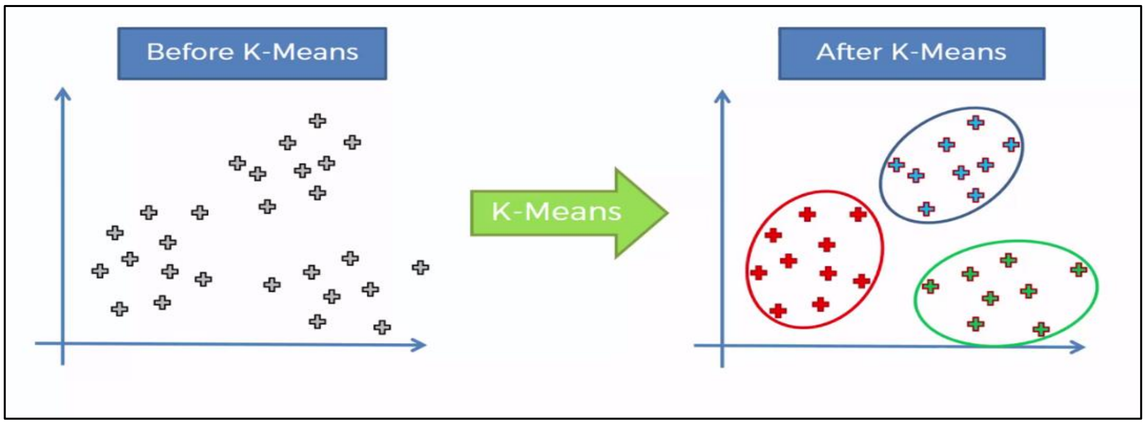


Figure 35: Depiction of clustering using K-means

An important part of implementing the K-Means algorithm is the assignment of the value of “k”, which as mentioned before is the number of clusters that we need the algorithm to provide us with. Since this thesis is utilising the HSL colour wheel, it can be determined by just manual observation that the colours can be clustered between 10-15 groups. But it is important to follow the proper procedure to determine the optimal value of “k”. This is done by implementing the elbow method. This is a plot of values of K on the x-axis and the distortion on the y-axis. The elbow chart for our particular use case looks as below.

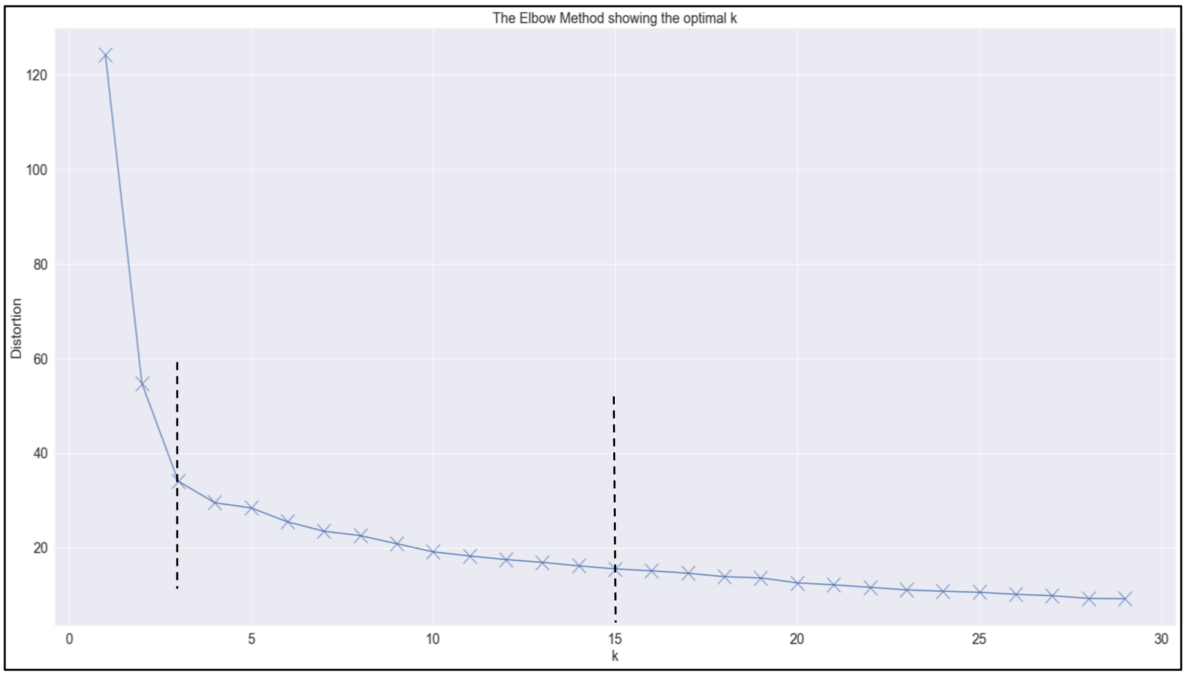


Figure 36: Elbow method for K-means to determine the optimum value for “k”

Here, from the initial observation, we see that the sharp elbow is located at around k value 3. But, from our initial observation from the HSL colour wheel and upon closer inspection at the subtle variations between 10 and 20, the value of “k” was set to 15. The

```
# dataset used to fit the model
_k_means_fit = prod_col[['hue','saturation','lightness']].values

# k means determine k
distortions = []
K = range(1,30)  # range set for number of clusters
for k in K:
    kmeanModel = KMeans(n_clusters=k).fit(_k_means_fit)
    kmeanModel.fit(_k_means_fit)
    distortions.append(sum(np.min(cdist(_k_means_fit, kmeanModel.clus-
ter_centers_, 'euclidean'), axis=1)) / _k_means_fit.shape[0])
```

Code Snippet 12: Python code to calculate the distortions for elbow chart for K-means elbow chart shown above was obtained by plotting the distortion measured for clusters between 1 and 30 which was calculated using the below python code.

Once the number of clusters was determined, the next part was to fit the model. This was done easily using the KMeans package from Scikit learn on python using the below code.

```python
# Assigning groups to products using K-Means

kmeans = KMeans(n_clusters=15, random_state=100).fit(_k_means_fit)
col_group = kmeans.labels_

# Create copy of original dataframe
prod_col_w_grp = prod_col.copy()
# Store generated groups into new column
prod_col_w_grp['group'] = list(col_group)
```

Code Snippet 13: Generation of groups for each product using K-Means

Once these groups were generated it was possible to visualise the clustering of the data points on a plot, which looks like below. Here each colour is a different cluster and the black points are the centres of each cluster also known as centroids.

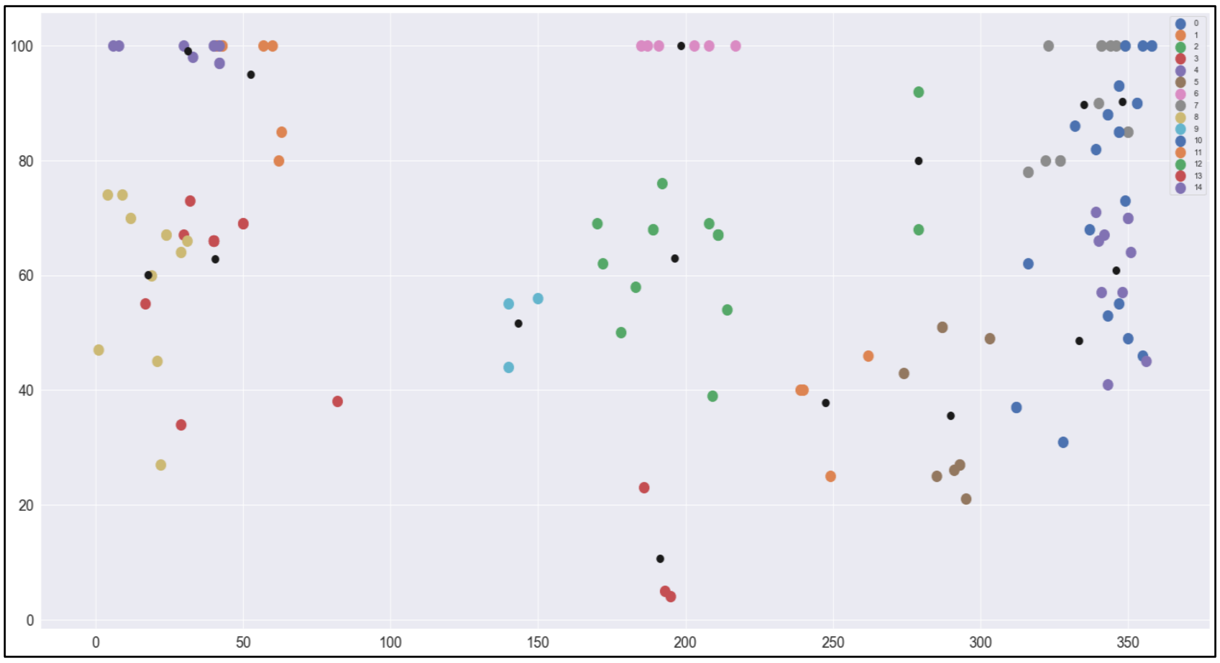


Figure 37: Clusters created on the dataset after using K-means

4.4 Categorisation based on DBSCAN algorithm

A lot of the most used clustering algorithms are based on the assumption that the data is generated based on a probability distribution of a given type. This is the case for the k-means algorithm described in the previous chapter. This assumption leads to the algorithms producing clusters that are spherical in nature even for datasets that might contain non-spherical structures. With the ever-increasing size of datasets, it is important to note this disadvantage of such algorithms and keeping this in mind, density-based clustering algorithms were developed. Such types of clustering do not make any assumptions related to the distribution of data or the number of clusters it might contain. Due to this fact, density-based clusterings can be treated as non-parametric methods (Aggarwal 2015).

Density-Based Spatial Clustering of Applications with Noise clustering or DBSCAN clustering is one such density-based clustering that can be used on datasets that have inconsistent shapes. This algorithm creates an estimation of the density by checking the total number of points within a radius (*Eps*) and considers the points as connected if they lie within the same neighbourhood. The clusters generated by this algorithm consists of the following components, a core point, the radius (*Eps*), a minimum number of points (MinPts) and noise. A point in the data set is considered as a core point if the neighbourhood of radius *Eps* contains at least the minimum number of set points. Keeping these basic terms in mind, the following definitions make up the DBSCAN algorithm (Martin Ester et al. 1996).

i. Directly Density-reachable – A point p is directly density-reachable from a point q wrt. Eps, MinPts if
    a. $p \in N_{Eps}(q)$ and
    b. $|N_{Eps}(q)| \geq$ MinPts (core point condition)
ii. Density-reachable – A point *p* is *density-reachable* from a point *q* wrt. Eps and MinPts if there are a chain of points $p_1, \ldots, p_n$, $p_1=q$, $p_n=p$, such that p_{i+1} is directly density-reachable from p_i.
iii. Density-Connected – A point p is density-connected to a point q wrt. Eps and MinPts if there is a point o such that both, p and q are density-reachable from o wrt. Eps and MinPts.
iv. Clustering – If D is a database of points, a cluster C is generated wrt. the Eps and MinPts which would be a non-empty subset of D based on the below conditions.
    a. $\forall$ p, q: if $p \in C$ and q are density-reachable from p wrt. Eps and MinPts, then $q \in C$. This condition is called Maximality.
    b. $\forall$ p, $q \in C$: p is density-connected to q wrt. Eps and MinPts. This condition is called Connectivity.

v. Noise – Consider $C_1, \ldots, C_k$ be clusters of database D wrt. parameters Eps_i and $MinPts_i$ where $i = 1, 2, \ldots, k$. Noise is defined as the set of points in the database D not belonging to any cluster C_i.

Based on this definition the parameters Eps and MinPts is required by the DBSCAN algorithm to generate clusters for a given dataset. While MinPts is a threshold that can be a set number of points or the entire dataset and can be set based on the necessity, an optimal value of Eps must be determined. This value must be calculated separately as the algorithm is not able to determine this value on its own. This calculation is based on the following observation. The distance is calculated between a point and the nearest n-points. This is repeated for all the points in the dataset after which all the distances are sorted and then plotted. Next, the plot is observed to find a point where there is a more pronounced change between the points. This was achieved in python using the below code snippet.

```python
# Import realted packages
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors

# reduced dataframe for fitting
_dbs_fit = prod_col[['hue','saturation','lightness']].values

# Calculate distances between neighbouring points
neigh = NearestNeighbors(n_neighbors=2)
nbrs = neigh.fit(_dbs_fit)
distances, indices = nbrs.kneighbors(_dbs_fit)

# Plot graph to measure optimal epsilon
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.figure(figsize=(20,10))
plt.plot(distances)
```

Code Snippet 14: Plotting the graph to estimate Eps value

The result of the above code snippet looks as below and the Eps value is determined at the value where a sharp curvature is observed. In this case, although the first instance is close to 0, the next instance is considered which is around 18.

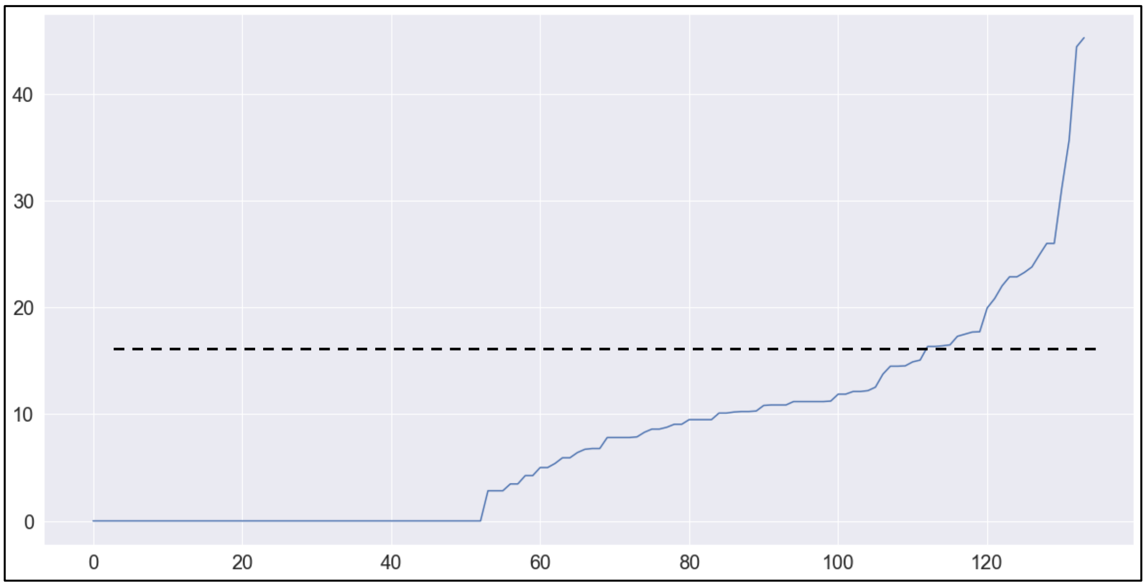


Figure 38: Plot determining the value of Eps

After the estimation of Eps, the MinPts needs to be determined. After manually checking the entire set of products, it was observed that there would be at least 2 products in each group and thus this value was considered as the MinPts parameter. After determining both the parameters necessary for the algorithm, the next step is to fit the model on our dataset. The Scikit Learn project contains a DBSCAN package which is utilised in this step and achieved easily using the below python code.

```python
## Fit the DBSCAN algorithm to the dataset
clustering = DBSCAN(eps=18, min_samples=2).fit(_dbs_fit)

# Generate the labels for the clusters
col_group_dbs = clustering.labels_

# Create copy of original dataset
prod_col_wdbs_grp = prod_col.copy()

# Append generated labels to the new dataframe
prod_col_wdbs_grp['group'] = list(col_group_dbs)
```

Code Snippet 15: Fit dataset to DBSCAN model and generate cluster labels

Once the cluster groups are generated, they can be easily visualised by plotting on a graph using the code snippet below.

```
# Plot clusters generated by DBSCAN

prod_dbs = prod_col_wdbs_grp[['hue','saturation','lightness', 'group']]
u_labels = np.unique(col_group_dbs)
colormap = np.array(['r', 'g', 'b']) # set the colour pallete of the col-
   ours for each group

plt.figure(figsize=(30,15)) # set size of the plot
# plotting the points for each group
for i in u_labels:
    plt.scat-
   ter(prod_dbs[col_group_dbs == i].iloc[:,0] , prod_dbs[col_group_dbs == i].iloc
   [:,1], s=200, label =i)
plt.legend()
plt.show()
```

Code Snippet 16: Visualising each cluster based on DBSCAN

In the below visualisation, all the clusters including the ones considered as noise (denoted by -1) can be observed. The values considered to be noise are the ones that do not fit in any of the other group. Based on the visual below, it can also be observed that even though a datapoint is very close to certain clusters, it is still considered as noise and is not mapped to that particular cluster. While the noise can be considered a group by itself, most of the data points within it might not be similar to each other. This can also be seen as a potential data loss considering how this group cannot be matched to any other group. It should also be noted that these points were assigned to clusters while using the K-Means algorithm as seen in Figure 37.

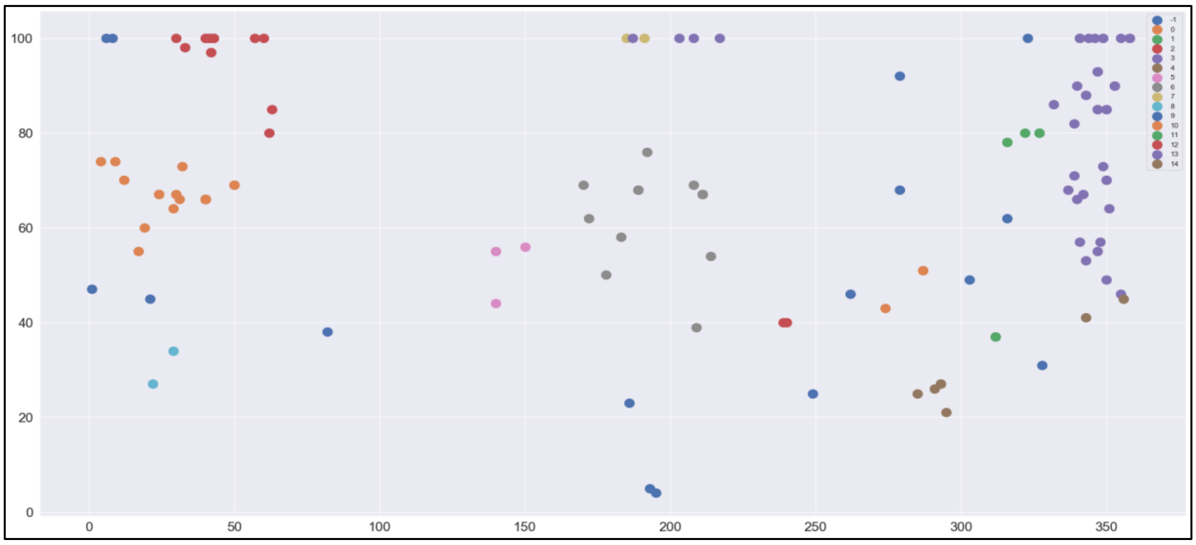


Figure 39: Plot showing different clusters including noise denoted by -1

This chapter has been an exploration of different possible approaches to cluster colours. Starting from a custom function where the hue, saturation and lightness values were manually split into groups based on how the angular distribution was of hues n the HSL colour wheel, we moved on to explore the Euclidean distances between colours. Both these functions returned all the products and were only filtered on the threshold of the number of products that were passed into the function. Although the function of Euclidean distance calculation did not necessarily cluster the products, what it did was return a dataset showing the products closest to the main product. The next two algorithms were the unsupervised machine learning models that generated clusters of the colours and "recommended" products could be easily searched by matching the corresponding label if the main product. After the implementation of the different models, the next step is to check the results. These results have been evaluated by two different methods, first, the visual similarity between the colours of the "recommended" products and second, a quantitative approach to determine which algorithm performed the clustering more efficiently. The next chapter describes these visual results and quantitative approaches to determine the best algorithm for this use case.

5 Exploring the results and determining the best clustering model

As described in Chapter 1, it is very difficult to measure the efficiency of clustering algorithms. This is because the algorithm decides the clustering based on its internal logic and the user would not have any reference to compare the result with. This makes it difficult to produce an evaluation metric that could be applied to all clustering algorithms uniformly and which could accurately determine the efficiency of the clustering.

For the clustering functions and algorithms that were described in chapter 5, there are a few approaches that were used to determine the better model. Since the total records are less, it is easy to check the visual differences by plotting the colours of an input product and comparing it with the colours of the resultant products. Apart from this basic approach, the Davies-Bouldin index was considered as a possible test to try and determine the best model. All the tests mentioned above are described below.

5.1 Determining the better models based on visual results

This test is conducted by generating a colour palette for the resultant products that were “recommended” or grouped for a particular input product. A sample product was considered as an input and fed into all the functions and models. The results for each of the functions and models are as below and they consist of the attributes and the colour of the input product and the same for the recommended products. The input product was set for “Toffee Delight” and the product looks as below. This image also shows how the product can look different based on the source of the image. The image on the left is the picture captured in a professional environment and the image on the right was clicked by a customer using the camera of their phone. This is the exact problem that we are trying to solve in this thesis.

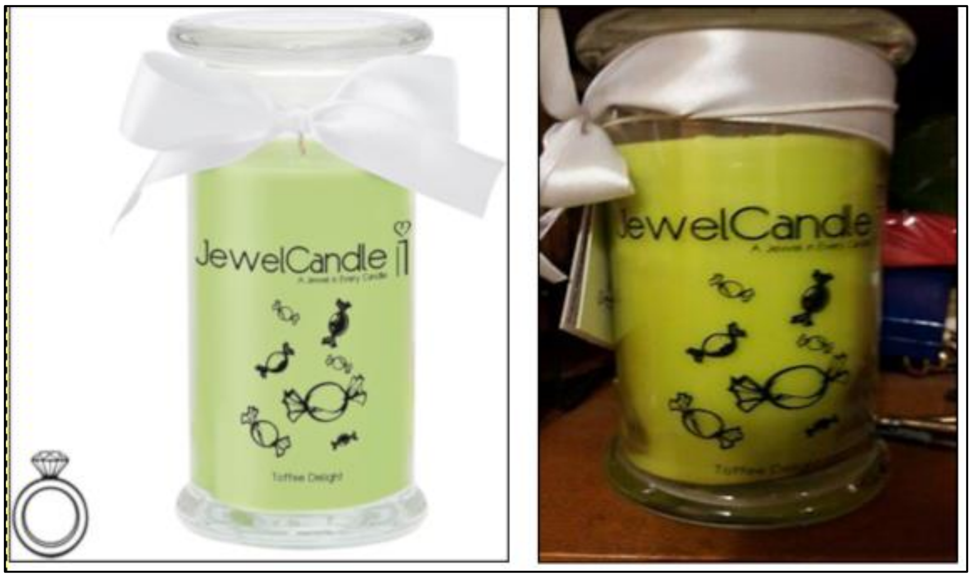
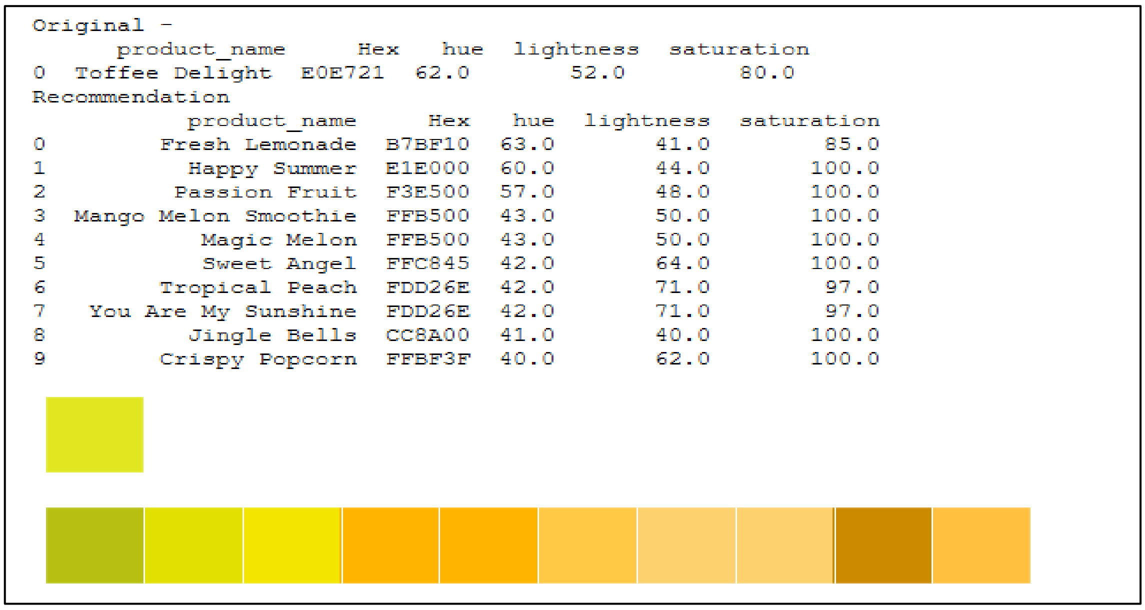


Figure 40: Images of Toffee Delight candle from two sources

The results are portrayed in the following format. In the colour palettes, the single colour is the colour of the input product and the palette underneath shows the colours for each of the recommendations generated by the function. The attributes for the recommendations are under the "*Recommendation*" section in the image below.

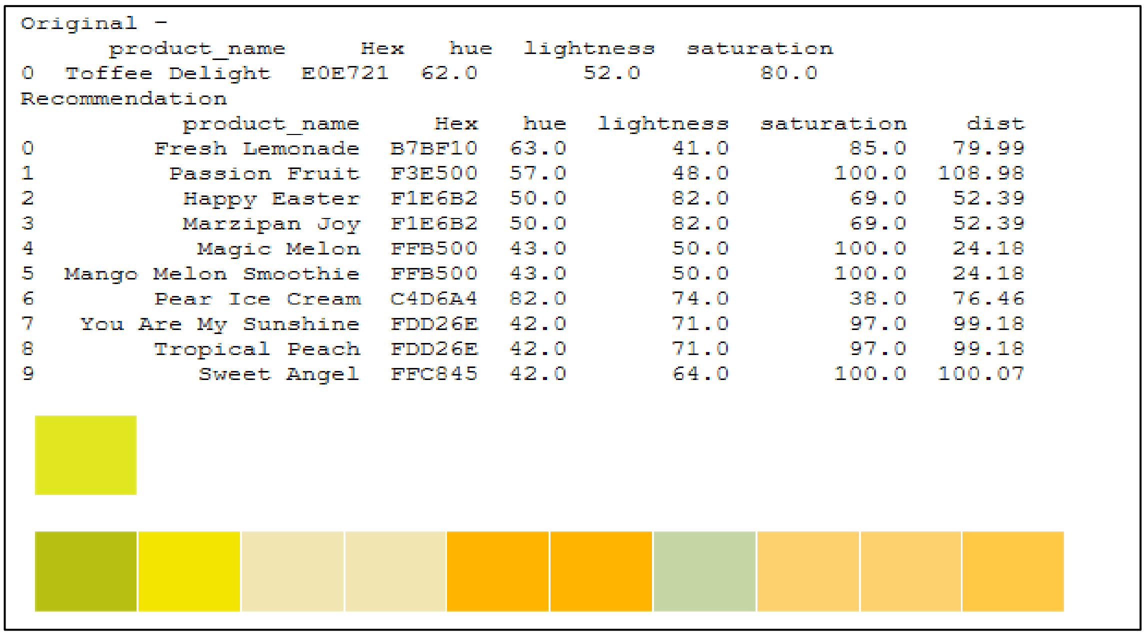
Figure 41 shows the results obtained from the custom function that was built by manually creating the groups for products based on the visual differences. This function splits the Hue, saturation and lightness attributes of the colours and creates groups of colours manually and compares these sub-groups with the same attributes of the main product. Then the function returns N number of products as required by the user based on the similarity of the products. This function is also resource-intensive as it performs the comparison between products every time and cannot be considered to be very efficient. As and when the dataset grows, the processing power would also increase and this is not an ideal case. This process has been described in chapter 4.1. From the results seen in figure 41, it can be seen that hue values of the "recommendations" are closer to the main product "Toffee Delight" and even visually the first three products are similar looking and gradually moving towards other products since there were no products with the closer hues.



```
Original -
      product_name     Hex    hue   lightness  saturation
0  Toffee Delight  E0E721  62.0        52.0         80.0
Recommendation
           product_name     Hex   hue  lightness  saturation
0        Fresh Lemonade  B7BF10  63.0       41.0        85.0
1          Happy Summer  E1E000  60.0       44.0       100.0
2         Passion Fruit  F3E500  57.0       48.0       100.0
3   Mango Melon Smoothie  FFB500  43.0       50.0       100.0
4           Magic Melon  FFB500  43.0       50.0       100.0
5           Sweet Angel  FFC845  42.0       64.0       100.0
6        Tropical Peach  FDD26E  42.0       71.0        97.0
7   You Are My Sunshine  FDD26E  42.0       71.0        97.0
8          Jingle Bells  CC8A00  41.0       40.0       100.0
9        Crispy Popcorn  FFBF3F  40.0       62.0       100.0
```

Figure 41: Results for clustering based on custom function described in Chapter 4.1

Figure 42 shows the results obtained from the function that was designed based on the Euclidean distances between the products. This mathematical function uses the Euclidean distance formula by integrating the hue, saturation and lightness attributes into the formula to calculate the distances between the main product and all other products and finally returns the top N products as required with the lowest distance between the main product and the other products. Similar to the custom function, this mathematical function also calculates the distance every time a “recommendation” is required. Although there are no manual groupings of the colours, the fact that the entire function runs on the entire dataset is not efficient. And again being similar to the custom function in terms of execution, this process would keep consuming more resources as and when the dataset increases. This function has been described in chapter 4.2. From the results seen in figure 41, it can be seen that the hue values of the “recommendations” are not as close to the main product as it was seen in the previous results. It can also be seen that the product “Happy Summer” is not present in these result which was available in the results of the custom function. We also see the product “Pear Ice Cream”, which does not fit in the colour scheme of the recommendations. These are a couple of key observations while comparing the results from the custom function and the mathematical function.

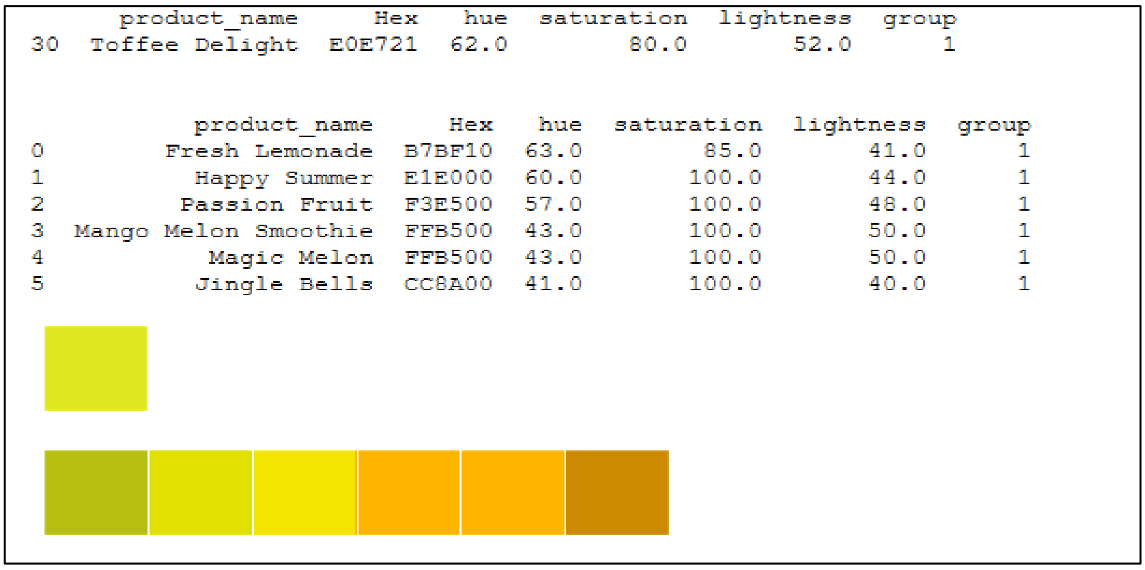


```
Original -
     product_name     Hex   hue  lightness  saturation
0  Toffee Delight  E0E721  62.0       52.0        80.0
Recommendation
          product_name     Hex   hue  lightness  saturation    dist
0        Fresh Lemonade  B7BF10  63.0       41.0        85.0   79.99
1         Passion Fruit  F3E500  57.0       48.0       100.0  108.98
2          Happy Easter  F1E6B2  50.0       82.0        69.0   52.39
3          Marzipan Joy  F1E6B2  50.0       82.0        69.0   52.39
4           Magic Melon  FFB500  43.0       50.0       100.0   24.18
5  Mango Melon Smoothie  FFB500  43.0       50.0       100.0   24.18
6        Pear Ice Cream  C4D6A4  82.0       74.0        38.0   76.46
7     You Are My Sunshine  FDD26E  42.0       71.0        97.0   99.18
8        Tropical Peach  FDD26E  42.0       71.0        97.0   99.18
9          Sweet Angel  FFC845  42.0       64.0       100.0  100.07
```

Figure 42: Results for the clustering function based on Euclidean distance

The previous two functions returned a set number of “recommendations” rather than identifying the clusters and tagging each product to its respective cluster. There is also the fact that the algorithms have to run every time a “recommendation” is needed and on an ever-increasing portfolio of products. This also makes these algorithms inefficient.

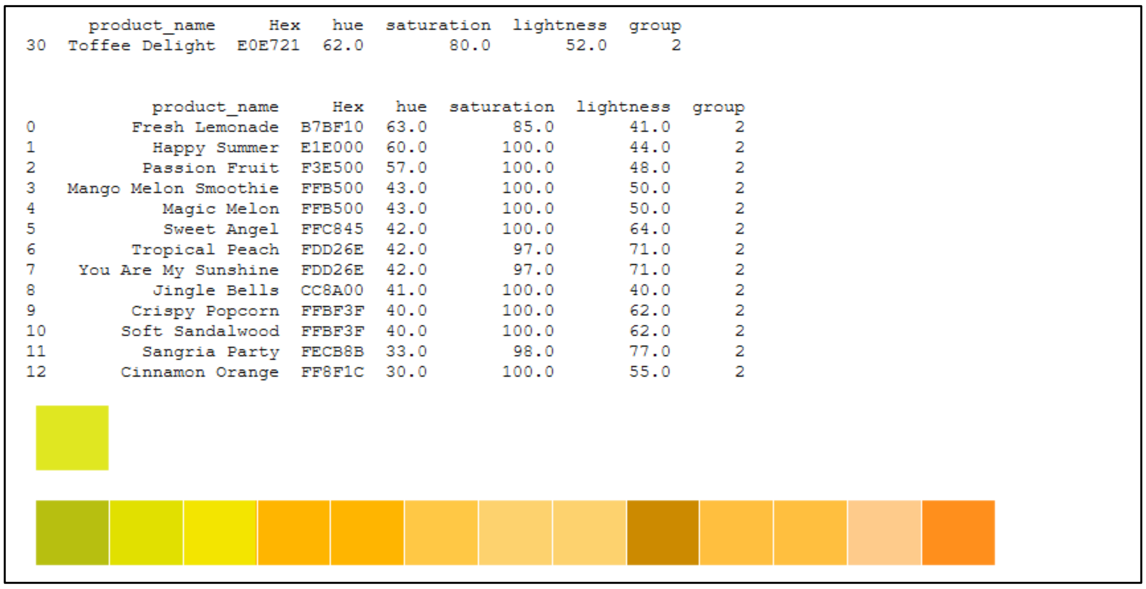
Figure 43 shows the results obtained from the clustering generated by using the K-Means algorithm. The K-Means algorithm is one of the most widely used clustering algorithms in the unsupervised machine learning domain. Unsupervised machine learning algorithms are helpful when there are no “labels” to identify and cluster different attributes. These algorithms create labels programmatically based on their internal logic and these labels can be used to identify clusters within a dataset. This makes it much easier and more efficient since we would only have to perform a clustering when there are new products available in the database and then use the labels for the “recommendations”. This function has been described in chapter 4.3. Here we can also see the groups that were assigned to the products while clustering. One thing to keep in mind with this result is that unlike the previous two functions where we can set the number of required recommendations, this algorithm clusters and label the datasets and only the products having the same label are returned. Visually, it can be seen that the results from the K-Means algorithm are very similar to the ones from the custom function.



```
        product_name      Hex    hue  saturation  lightness  group
30    Toffee Delight   E0E721   62.0        80.0       52.0      1


                 product_name     Hex    hue  saturation  lightness  group
0              Fresh Lemonade  B7BF10   63.0        85.0       41.0      1
1                Happy Summer  E1E000   60.0       100.0       44.0      1
2               Passion Fruit  F3E500   57.0       100.0       48.0      1
3         Mango Melon Smoothie  FFB500   43.0       100.0       50.0      1
4                 Magic Melon  FFB500   43.0       100.0       50.0      1
5                Jingle Bells  CC8A00   41.0       100.0       40.0      1
```

Figure 43: Results of clustering based on the K-Means algorithm

Figure 44 shows the results obtained from the clustering generated by using the DBSCAN algorithm. Being one of the two unsupervised clustering algorithms used, it is important to compare the results of DBSCAN with K-Means. This function has been described in chapter 4.4. Here we can also see the groups that were assigned to the products while clustering. When compared to K-Means, we see that DBSCAN has more products clustered together. And they are similar visually as well and from first look this seems to be a better model for our use case. But it must also be kept in mind that DBSCAN clusters together all the data points that could be clustered into other groups. This causes "wastage" since those products would not show up as recommendations.



```
        product_name     Hex   hue  saturation  lightness  group
30    Toffee Delight  E0E721  62.0        80.0       52.0      2


                product_name     Hex   hue  saturation  lightness  group
0             Fresh Lemonade  B7BF10  63.0        85.0       41.0      2
1               Happy Summer  E1E000  60.0       100.0       44.0      2
2              Passion Fruit  F3E500  57.0       100.0       48.0      2
3       Mango Melon Smoothie  FFB500  43.0       100.0       50.0      2
4                Magic Melon  FFB500  43.0       100.0       50.0      2
5                Sweet Angel  FFC845  42.0       100.0       64.0      2
6             Tropical Peach  FDD26E  42.0        97.0       71.0      2
7        You Are My Sunshine  FDD26E  42.0        97.0       71.0      2
8               Jingle Bells  CC8A00  41.0       100.0       40.0      2
9             Crispy Popcorn  FFBF3F  40.0       100.0       62.0      2
10           Soft Sandalwood  FFBF3F  40.0       100.0       62.0      2
11             Sangria Party  FECB8B  33.0        98.0       77.0      2
12           Cinnamon Orange  FF8F1C  30.0       100.0       55.0      2
```

Figure 44: Results of clustering based on the DBSCAN algorithm

Now that the results from the various experimentations have been collected, the next step would be to shortlist the most prominent clustering algorithms and finally decide on the best algorithm that can be used for the use case in focus. This can be done in two steps.

Step 1: Shortlisting algorithms based on visual results.

The image below collates the colour palettes of the products that were presented as “recommendations” or that were clustered together by the different functions and algorithms that were discussed in chapter 5. This will be of assistance to form a first impression of the results and to shortlist some algorithms or functions which would then need further inspection. The below image consists of the colour of the input product and the colour palettes generated for the products that were considered similar to the input product by the different functions and algorithms.

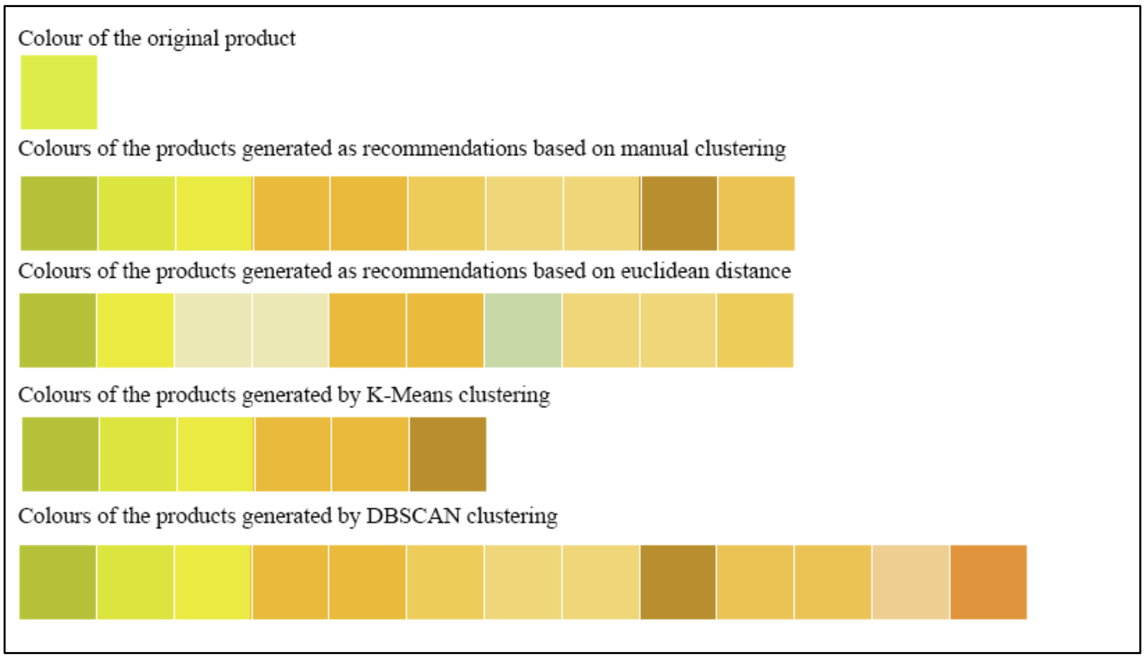


Figure 45: Results of clustering algorithms and functions

By observing figure 45, it can be seen that all the algorithms and functions that were experimented with produced more or less similar results. The main difference that can be seen at first sight is the difference in the number of results. For the first two functions that were based on manual clustering and identifying similar products based on Euclidean distances, we see ten products since it was set in the function by default. Upon closer inspection for the clustering based on Euclidean distances, we can see that some colours don’t match or are similar to the colour of the input product as well as the other functions and algorithms. There is also the fact of the inefficiency of needing these algorithms to run every time on the entire data set to calculate the similarity and distances whenever a visit to the website would be made. As a result, this function can be excluded from the consideration of the best model. Excluding this result, the remaining three clusterings are very similar. Now, it must be taken into notice that the first algorithm described in chapter 4.1 is not perfect and has a lot of flaws. One of the main drawbacks is the need to manually sort and cluster each product, which would have to be done every time a new product is added into the database. Apart from this drawback, the logic to compare the attributes of the input with each product in the data set every time clustering is required affects the efficiency of the function. Due to these reasons, this function can also be excluded. This leaves us with two well-known clustering algorithms that are widely used in the Data Science community. Since the two clustering algorithms are now shortlisted, the next step is to check the efficiency of the algorithms and finally decide on one of them as the main model to cluster our dataset.

Step 2: Determine the most efficient model based on Davies Bouldin scores

After shortlisting the two clustering algorithms, the next step is to determine a metric value that could potentially be used to determine the better model. Davies-Bouldin score is one such metric that tests the efficiency of clustering algorithms. This test is described in the next chapter.

5.2 Davies-Bouldin index score for K-Means and DBSCAN clustering

David L. Davies and Donald W. Bouldin presented a measure in 1979 that could be used to "*indicate the similarity of clusters which are assumed to have a data density which is a decreasing function of distance from a vector characteristic of the cluster.*" (Davies and Bouldin 1979)

This function was created keeping the below criteria in mind.

1. It should require little to no user interaction or specification of parameters.
2. It should apply to hierarchical datasets.
3. It should be computationally feasible for relatively large datasets.
4. It should yield meaningful results for data in arbitrary dimensionality.

The objective of this score is to calculate the average similarity of each cluster with another cluster that is the most similar to it. Consider the following variables,

C_i is a cluster where i is in the range of 1, …, k where k is the total number of clusters and the most similar cluster to this cluster is C_j. For this particular scenario, the average similarity can be denoted as R (S_i, S_j, M_{ij}), where M_{ij} is the distance between the vectors of clusters i and j, and S_i and S_j are the dispersions of clusters i and j respectively. Considering these parameters, the simplest function to define a nonnegative and symmetric score is defined by the formula below.

$$R_{ij} = \frac{s_i + s_j}{M_{ij}}$$

And based on this formula the Davies-Bouldin score is calculated using the formula below.

$$\bar{R} \equiv \frac{1}{k} \sum_{i=1}^{N} R_i$$

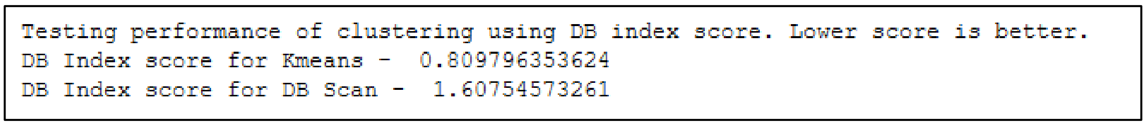
Where $R_i \equiv$ maximum of R_{ij} $i \neq j$.

It is important to know that the closer the score is to non-negative zero, the better is the clustering. Keeping this in mind, we can also say that while comparing two scores, the clustering with the lower score is the better one compared to the other (Tempola and Assagaf 2018).

This concept has been implemented in the Sci-kit library as davies_bouldin_score and requires two parameters, viz. the data set that was fit into the model and the labels generated through the clustering model. The below snippet shows the implementation and then the scores for K-Means and the DBSCAN algorithms.

```
from sklearn.metrics import davies_bouldin_score
## lower score is better
## _k_means_fit is the dataset used to fit the K-Means model and col_groups is the la-
     bels generated with the K-Means algorithm
## _dbs_fit is the dataset used to fit the DBSCAN model and col_groups_dbs is the la-
     bels generated with the DBSCAN algorithm

print('Testing performance of clustering using DB index score. Lower score is bet-
     ter.')
print('DB Index score for Kmeans - ', davies_bouldin_score(_k_means_fit, col_group))
print('DB Index score for DB Scan - ', davies_bouldin_score(_dbs_fit, col_group_dbs))
```

Code Snippet 17: Code snippet to generate and print the Davies-Bouldin score for K-Means and DBCSAN clustering results



```
Testing performance of clustering using DB index score. Lower score is better.
DB Index score for Kmeans -  0.809796353624
DB Index score for DB Scan -  1.60754573261
```

Figure 46: Results for Davies-Bouldin scores.

The below table gives the descriptions of the parameters used to generate the scores.

Parameter	Description
`_k_means_fit`	Data set used to fit the K-Means clustering algorithm
`col_group`	Labels generated by the K-Means clustering algorithm
`_dbs_fit`	Data set used to fit the DBSCAN clustering algorithm
`col_group_dbs`	Labels generated by the DBSCAN clustering algorithm

Table 2: Description of parameters to generate Davies-Bouldin score

Starting from designing custom function by manually creating groups for colours and implementing a mathematical model to calculate the distances between attributes to using unsupervised machine learning models to clusters the colours, there have been differences in performance, efficiency and the generated results that were observed. Out of the four algorithms experimented with we shortlisted the two ML models based on the fact that the custom and mathematical functions were inefficient to work with due to the requirement of constantly running them even though the visual results were not bad and there was a feature to retrieve as many “recommendations” as needed. Continuing with the two ML models, the visual results were similar except for the fact that DBSCAN had more results for the example we had considered for testing. There can be instances where K-Means would generate more recommendations compared to DBSCAN and so it was necessary to determine the efficiency of the clustering quantitatively which was achieved with the help of the Davies-Bouldin index which determines the more efficient clustering when comparing different clustering models. From the results of the Davies-Bouldin score, we can infer that the K-Means model has performed a better clustering than DBSCAN for the use case explored in this thesis. Based on these results and the fact that DBSCAN has “wastage” while clustering we can determine that the K-Means clustering works the best for the data set we have at hand and can be continued as the ideal solution for the clustering of colours for our particular use case.

6 Final Conclusion and further development

Clustering is a problem that is prevalent in many different scenarios and the problem described in this thesis was one of them. Something that seemed to be a difficult task if someone manually goes through the entire inventory of the company to sort and label each product based on its looks could be solved with the help of data science and unsupervised machine learning algorithms. And with the exploration conducted in this thesis, it can be seen that it need not be a complicated and cumbersome task provided that the data is available or can be modified to obtain the required format. The most important fact that needs to be considered while building such solutions is the willingness to explore beyond conventional thinking because useful information can be obtained from the most unexpected sources.

This project involved the exploration of various topics such as human anatomy, structures and formats of colours and finally different approaches to tackling the clustering problem. It is also important to understand that not all data is readily available from a single source and it might be required to extract necessary data from sources that might sometimes be considered unconventional. These are important factors that need to be considered while building a data science solution.

Starting from chapter 1 where the motivation for this project was introduced along with a brief understanding of data science concepts, we moved on to chapter 2 where the human visual system and human perception of colours were looked into. Along with human biology that is a key factor in this analysis, chapter 2 also goes through the concept of colours, different colour models and then finally decides on the HSL model that would be used for this analysis after comparing the other models. Chapter 3 focused on the data preparation aspect of this analysis by extracting relevant fields from the product database in the company backend and manipulating the Pantone codes to convert them into hue, saturation and lightness values in the HSL colours space. After preparing the dataset we moved to chapter 4 where the different potential functions and machine learning models were described and experimented on. And chapter 5 evaluated the results from the experimentations by comparing the visual results which lead to shortlisting the K-Means and DBSCAN model and finally performing a quantitative test to determine which model among K-Means and DBSCAN performed a better clustering.

Based on the tests that were conducted, it can be safely assumed that K-Means is a good model to use for the clustering of the data set. Considering the low volume of the data the model also runs efficiently without any issues. Although the other algorithms and functions worked well it is important to note the disadvantages they had over K-Means. These ranged from being inefficient to creating wasted clusters that could not be utilised. There are many more methods that could be used to cluster data which could be much more efficient and might provide even better results. But from the analysis and experimentations carried out in this thesis, K-Means has proved to be an effective starting point and can be used as a base model for further development and expansion of this project. Using this solution, the products can now be clustered efficiently and then later be integrated with the online shop. This along with the integration of a function to identify products with lower sales will help in achieving the final goal of cross-selling and also promoting the sales of other products that were otherwise sold less compared to the most popular products in the store.

Building on top of the solution presented, there is scope for future development and expansion of the problem statement. While unsupervised machine learning algorithms is one approach to this problem, another approach could be using image recognition algorithms and deep learning to determine the similarity between products. The clustering problem described in this thesis can be further expanded upon by using attributes other than colour for clustering. This thesis focuses on clustering only the colour attribute of the candle and the next step would be to cluster the smells and finally combine the results to find the products that have the most correlation between each other when compared to another product. This would be the basic idea that would be worked on in the future. This analysis would touch on another aspect of human biology which would be the sense of smell as well as exploring the psychology behind preferring one scent compared to another. Smell being something that cannot be detected by a computer unless special sensors or products such as an electronic nose can be expensive and would be deemed as an unnecessary expense. The approach to achieve this would be using the ingredients used in a particular scent and the structure of the scent and using these attributes for clustering. This analysis can be achieved with the help of a combination of natural language processing and clustering algorithms such as K-Means. These analyses can be explored further in the future depending on the necessity and the availability of resources.

List of literature

1. Ford, A. and Roberts, A. (1998), *Colour space conversions*.
2. Gray, H. (1878), *Anatomy of the Human Body*, Lea & Febiger.
3. German Institute for Standardization (2017), "DIN 5033-1. Colorimetry - Part 1: Basic terms of colorimetry".
4. Hunt, R.W.G. and Pointer, M. (2011), *Measuring colour, Wiley-IS&T series in imaging science and technology*, 4th ed., Wiley, Chichester, West Sussex, U.K.
5. Tkalcic, M. und Tasic, J.F. (2003), "Colour spaces: perceptual, historical and applicational background", in Zajc, B. (Hrsg.), *Computer as a tool*, IEEE.
6. Nishad PM (2013), "VARIOUS COLOUR SPACES AND COLOUR SPACE CONVERSION", *Journal of Global Research in Computer Science*, Vol. 4 No. 1, ff. 44-48.
7. Fairchild, M.D. (2005), *Color appearance models, Wiley-IS & T series in imaging science and technology*, 2nd ed., J. Wiley, Chichester, West Sussex, England, Hoboken, NJ.
8. Cotton, S.D.'O. (1995), *Colour, colour spaces and the human visual system, School of Computer Science, University of Birmingham*.
9. MacEvoy, B. (2010), "Color Vision", Available at: https://www.handprint.com/LS/CVS/color.html (Accessed on 19. July 2020).
10. Pointer, M.R. (1981), "A comparison of the CIE 1976 colour spaces", *Color Research & Application*, Vol. 6 No. 2, ff. 108-118.
11. Cochrane, S. (2014), "The Munsell Color System: a scientific compromise from the world of art", *Studies in History and Philosophy of Science Part A*, Vol. 47, ff. 26-41.
12. A.H. Munsell (1907), *A Color Notation: A measured color system, based on the three qualities Hue, Value and Chroma*, Geo. H. Ellis Co, Boston.
13. Wikipedia (2021), "Munsell color system", Available at: https://en.wikipedia.org/w/index.php?title=Munsell_color_system&oldid=1006856810 (Accessed on 10. May 2020).
14. Munsell Color System and Color Matching from Munsell Color Company (2011), "Munsell Chroma; 3 Dimensions of Color | Munsell Color System; Color Matching from Munsell Color Company", Available at: https://munsell.com/about-munsell-color/how-color-notation-works/munsell-chroma/ (Accessed on 8. May 2020).

15. Munsell Color System and Color Matching from Munsell Color Company (2011), “Munsell Value Scale; 3 Dimensions of Color | Munsell Color System; Color Matching from Munsell Color Company”, Available at: https://munsell.com/about-munsell-color/how-color-notation-works/munsell-value/ (Accessed on 8. May 2020).
16. Munsell Color System and Color Matching from Munsell Color Company (2011), “Munsell Hue; 3 Dimensions of Color | Munsell Color System; Color Matching from Munsell Color Company”, Available at: https://munsell.com/about-munsell-color/how-color-notation-works/munsell-hue/ (Accessed on 8. May 2020).
17. Munsell Color System and Color Matching from Munsell Color Company (2011), “Munsell Color Space & Solid | Munsell Color System; Color Matching from Munsell Color Company”, Available at: https://munsell.com/about-munsell-color/how-color-notation-works/munsell-color-space-and-solid/ (Accessed on 8. May 2020).
18. Munsell Color System and Color Matching from Munsell Color Company (2011), “Development of the Munsell Color Order System | Munsell Color System; Color Matching from Munsell Color Company”, Available at: https://munsell.com/about-munsell-color/development-of-the-munsell-color-order-system/ (Accessed on 8. May 2020).
19. Nikolai Waldman (2013), “Math behind colorspace conversions, RGB-HSL – Niwa”, Available at: http://www.niwa.nu/2013/05/math-behind-colorspace-conversions-rgb-hsl/ (Accessed on 8. May 2020).
20. “PMS to Hex Color Chart | Pantone Color to Hex Conversion” (2020), Available at: https://www.easycalculation.com/colorconverter/pantone-to-hex-table.php (Accessed on 8. May 2020).
21. “History of Pantone Inc. – FundingUniverse” (2021), Available at: http://www.fundinguniverse.com/company-histories/pantone-inc-history/ (Accessed on 2. May 2020).
22. Pantone (2020), “Fashion, Home + Interiors Color Guide Book (FHIP110A) | Pantone”, Available at: https://www.pantone.com/fashion-home-interiors-color-guide (Accessed on 2. May 2020).
23. Wikipedia (2020), “Color”, Available at: https://en.wikipedia.org/w/index.php?title=Color&oldid=955495008 (Accessed on 8. May 2020).
24. Wikipedia (2020), “HSL and HSV”, Available at: https://en.wikipedia.org/w/index.php?title=HSL_and_HSV&oldid=955385375 (Accessed on 8. May 2020).

25. Codrops, T. (2016), “hsl | Codrops”, Available at: https://tympanus.net/codrops/css_reference/hsl/ (Accessed on 8. May 2020).
26. “linear algebra - How come that HSL can contain more information than RGB? - Mathematics Stack Exchange” (2020), Available at: https://math.stackexchange.com/questions/1033671/how-come-that-hsl-can-contain-more-information-than-rgb (Accessed on 8. May 2020).
27. Wikipedia (2020), “Color vision”, Available at: https://en.wikipedia.org/w/index.php?title=Color_vision&oldid=954194573 (Accessed on 10. May 2020).
28. Wikipedia (2020), “Trichromacy”, Available at: https://en.wikipedia.org/w/index.php?title=Trichromacy&oldid=950821801 (Accessed on 10. May 2020).
29. Hulshof, J.J.M. and Gehring, F.C., Koninklijke Philips NV, 2003. *Display device and cathode ray tube*. U.S. Patent 6,528,958.
30. *HowStuffWorks* (2000), “How Computer Monitors Work” 16 June, Available at: https://computer.howstuffworks.com/monitor7.htm (Accessed on 10. May 2020).
31. Wikipedia (2020), “CMYK color model”, Available at: https://en.wikipedia.org/w/index.php?title=CMYK_color_model&oldid=951230732 (Accessed on 10. May 2020).
32. Wikipedia (2020), “Color space”, Available at: https://en.wikipedia.org/w/index.php?title=Color_space&oldid=951382780 (Accessed on 10. May 2020).
33. Wikipedia (2020), “RGB color model”, Available at: https://en.wikipedia.org/w/index.php?title=RGB_color_model&oldid=943849582 (Accessed on 10. May 2020).
34. Mohri, M., Rostamizadeh, A. and Talwalkar, A. (2018), *Foundations of Machine Learning, Adaptive Computation and Machine Learning Series,* Second edition, The MIT Press, Cambridge, MA.
35. Rüdiger Wirth and Jochen Hipp (2000), *CRISP-DM: Towards a standard process model for data mining.*
36. Nicholas Njiru and Elisha Opiyo (2018), *Clustering and Visualizing the Status of Child Health in Kenya: A Data Mining Approach*.
37. Van Otterlo, M. and Wiering, M., (2012). *Reinforcement learning and markov decision processes*.
38. “SUPERVISED MODELS | Data Vedas” (2021), Available at: https://www.datavedas.com/supervised-models/ (Accessed on 8. January 2021).
39. “UNSUPERVISED MODELS | Data Vedas” (2021), Available at: https://www.datavedas.com/unsupervised-models/ (Accessed on 8. January 2021).

40. Fred Safier (2003), *Schaum's outline of theory and problems of precalculus,* McGraw-Hill, Year: 1997.
41. Steinley, D. (2006), “K-means clustering: a half-century synthesis”, *The British journal of mathematical and statistical psychology,* Vol. 59 Pt 1, ff. 1-34.
42. Aggarwal, C.C. (2015), *Data mining: The textbook / Charu C. Aggarwal*, Springer, Cham.
43. Davies, D.L. und Bouldin, D.W. (1979), “A Cluster Separation Measure”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1 No. 2, ff. 224-227.
44. Tempola, F. and Assagaf, A.F. (2018), “Clustering of Potency of Shrimp In Indonesia With K-Means Algorithm And Validation of Davies-Bouldin Index”, *International Conference on Science and Technology (ICST 2018)*, ff. 730-733.
45. Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu (1996), *A density-based algorithm for discovering clusters in large spatial databases with noise,* 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96).
46. Rahmah, N. und Sitanggang, I.S. (2016), “Determination of Optimal Epsilon (Eps) Value on DBSCAN Algorithm to Clustering Data on Peatland Hotspots in Sumatra”, *IOP Conference Series: Earth and Environmental Science*, Vol. 31 No. 1, f. 12012.
47. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot und Édouard Duchesnay (2011), “Scikit-learn: Machine Learning in Python”, *Journal of Machine Learning Research*, Vol. 12 No. 85, ff. 2825-2830.
48. Rehman, S.U., Asghar, S., Fong, S. und Sarasvady, S. (2014), “DBSCAN: Past, present and future”, in *Applications of Digital Information and Web Technologies (ICADIWT), 2014 Fifth International Conference on the* IEEE.
49. Pham, D.T., Dimov, S.S. und Nguyen, C.D. (2005), “Selection of K in K -means clustering”, *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, Vol. 219 No. 1, ff. 103-119.
50. Helmholtz, H. von (2013), *Treatise on Physiological Optics, Volume III, Dover Books on Physics*, v. 3, Dover Publications, Newburyport.
51. Thompson, P. (1985), “Visual Perception: an Intelligent System with Limited Bandwidth”, *Fundamentals of Human–Computer Interaction*, ff. 5-33.

52. Barbur, J.L. und Rodriguez-Carmona, M. (2012), “Variability in normal and defective colour vision: consequences for occupational environments”, *Colour Design,* ff. 24-82.
53. Shevell, S.K. (2003), *The science of color,* 2nd ed., Elsevier; [United States] Optical Society of America, Amsterdam, Boston.
54. Bueno, G., González, R., Déniz, O., González, J. und García-Rojo, M. (2008), “Colour model analysis for microscopic image processing”, *Diagnostic Pathology,* 3 Suppl 1 Suppl 1, S18.
55. Koschan, A. (2008), *Digital Color Image Processing [electronic resource],* Wiley.
56. Plataniotis, K.N. and Venetsanopoulos, A.N. (20), *Color Image Processing and Applications,* Springer International Publishing, Cham.
57. Fiete, R.D. (2007), *Image Chain Analysis for Space Imaging Systems*, Vol. 51, Society for Imaging Science and Technology.
58. Abdul Nasir, A.S., Mashor, M.Y. und Rosline, H. (2011), “Unsupervised colour segmentation of white blood cell for acute leukaemia images”, in Staff, I. (Hrsg.), *2011 IEEE International Conference on Imaging Systems and Techniques,* IEEE, [Place of publication not identified].
59. Buchsbaum, G. und Gottschalk, A. (1983), “Trichromacy, opponent colours coding and optimum colour information transmission in the retina”, *Proceedings of the Royal Society of London. Series B, Biological sciences,* Vol. 220 No. 1218, ff. 89-113.
60. Jordan, M.I. and Mitchell, T.M. (2015), “Machine learning: Trends, perspectives, and prospects”, *Science,* Vol. 349 No. 6245, ff. 255-260.
61. Richard S. Sutton (1992), “Introduction: The Challenge of Reinforcement Learning”, in *Reinforcement Learning,* Springer, Boston, MA, S. 1–3.
62. Abe, N., Biermann, A.W. und Long, P.M. (2003), “Reinforcement Learning with Immediate Rewards and Linear Hypotheses”, *Algorithmica,* Vol. 37 No. 4, ff. 263-293.
63. S. Singh, R. L Lewis and A. G Barto (2009), *Where Do Rewards Come From?*
64. Smith, A.R. (1978), “Color gamut transform pairs”, *ACM SIGGRAPH Computer Graphics,* Vol. 12 No. 3, ff. 12-19.
65. Miguel A. Teixeira, Oscar Rodríguez, Paula Gomes, Vera Mata and Alírio E. Rodrigues (2013), *Perfume Engineering: Design, Performance & Classification.*

66. Wikipedia (2021), “Reinforcement learning”, Available at: https://en.wikipedia.org/w/index.php?title=Reinforcement_learning&oldid=1009612279 (Accessed on 20. February 2021).
67. Deepmind (2021), “AlphaGo: The story so far”, Available at: https://deepmind.com/research/case-studies/alphago-the-story-so-far (Accessed on 20. February 2021).
68. Pantone (2021), “Find a Pantone Color | Quick Online Color Tool | Pantone”, Available at: https://www.pantone.com/color-finder (Accessed on 2. March 2021).
69. Wikipedia (2021), “CIE 1931 color space”, Available at: https://en.wikipedia.org/w/index.php?title=CIE_1931_color_space&oldid=1009083250 (Accessed on 5. March 2021).

Statutory Declaration

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper.

I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content.

I am aware that the violation of this regulation will lead to the failure of the thesis.

Abhishek Ravindra Kurup	Abhishek
Student's name	Student's signature
567493	05-03-2021
Matriculation number	Berlin, date