Clustering Data and Visualizing in 2D

Table of contents

1 Introduction

In this project, titled as *Clustering Data and Visualizing in 2D*, we aim to group similar data points of a dataset together and then visualize *where* and *how* those groups separate from each other and explore what characterizes them. From a theoretical point of view, we plan to explore consepts such as:

- General Data Science pipeline
- Unsupervised learning
- Representation learning
- High-dimensional data characteristics
- Neural networks,

while from the practical aspect, we expect to get hands on experience in using K-means and Self-Organizing Map (SOM) algorithms. The diagram below provides a high level view on the topics covered in the project and how they build upon each other.

Additionally, we will use 2D visualizations to **explore**, **understand** and **explain patterns** in the data. We will get hands on experience how unsupervised and representation learning works along with how to visualize high-dimensional data in a way that is interpretable for the human eye.

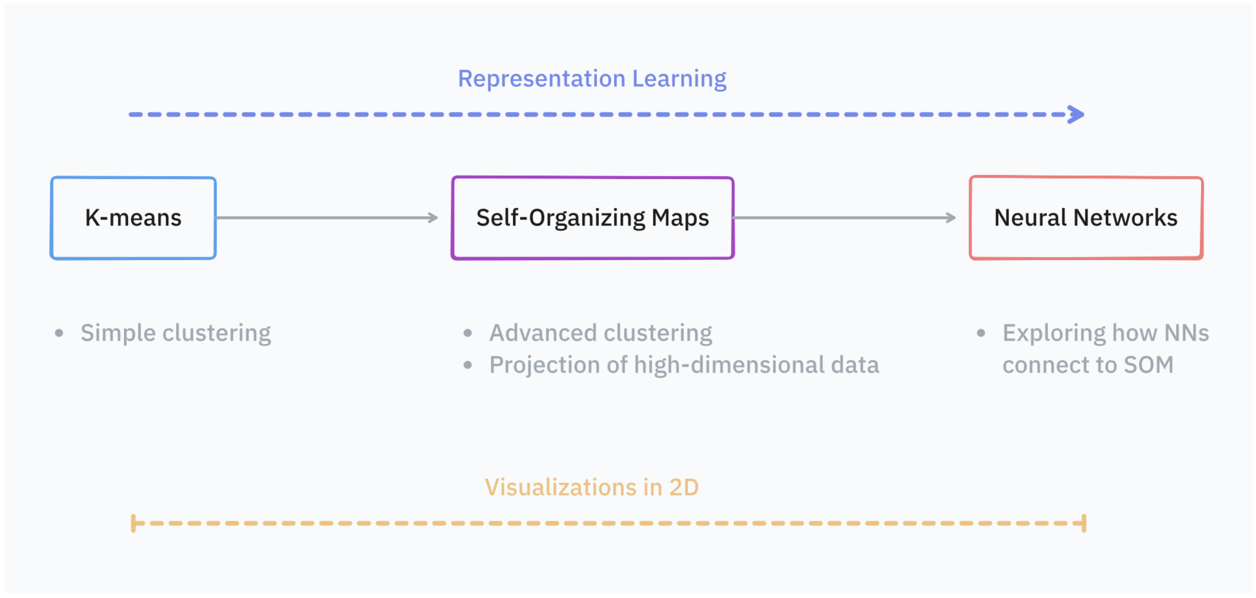


Figure 1: Bird's eye view of the project.

2 Background

2.1 General Data Science pipeline

Figure 2 depicts the general workflow of Data Science while Figure 3 displays a broader view on the process. Elaboration of the main steps are as follows.

Define and understand the problem

The problem needs to be clearly defined and then turned into a Data Science task, so clear steps can be defined on how to solve it.

Data collection

In this step, it is important to collect the data that is suitable for the job. It is a good idea to collect more data than you need as data sets can be incomplete and dirty. Although, it is very important to highlight that the *quality* of the data is more important than the *quantity*.

Data cleaning and preprocessing

Raw data can include missing entries, duplicates, extreme outliers to name a few. Un- or improperly processed data will lead to bad models.

Data exploration (EDA)

In this phase—often referred to as *Exploratory Data Analysis*—characteristics of the dataset are summarized and visualized. It helps in understanding what the collected data tells us: answer questions right away, see patterns or anomalies that could guide in building a better model.

Model building

In this part, the preprocessed data is used to train a model that discovers hidden patterns or learn from it in order to make predictions for the future.

Evaluation

The trained model is evaluated on never before seen samples that we refer to as the test (or holdout) set, by which we can verify how well the model generalizes.

Deployment and refinement

In this stage, models get deployed, observed and refinements are initiated to start the cycle again.

2.2 Supervision in Machine Learning

2.2.1 Supervised

In supervised feature learning, the input data is labelled, meaning data given to the model includes input-label pairs. This resorts in a representation with high label prediction accuracy. Examples include supervised neural networks, multilayer perceptrons, and dictionary learning.

2.2.2 Unsupervised

Unsupervised feature learning uses unlabelled input data, it learns features by analyzing the relationship between data points. Examples include Clustering (K-means), ICA and matrix factorization.

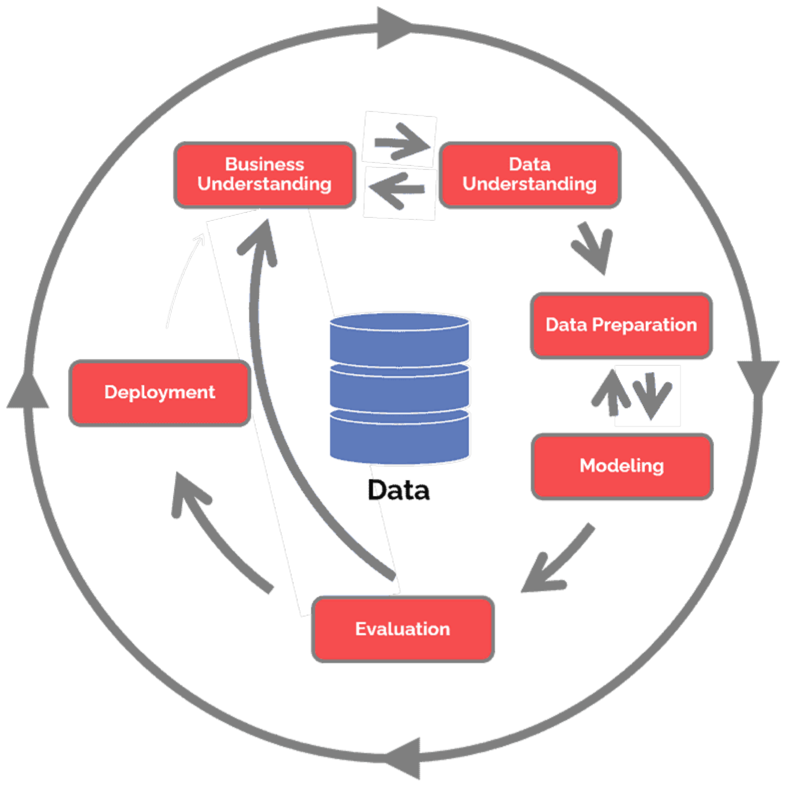


Figure 2: General pipeline of Data Science. Figure by DataScience-PM.com [1].

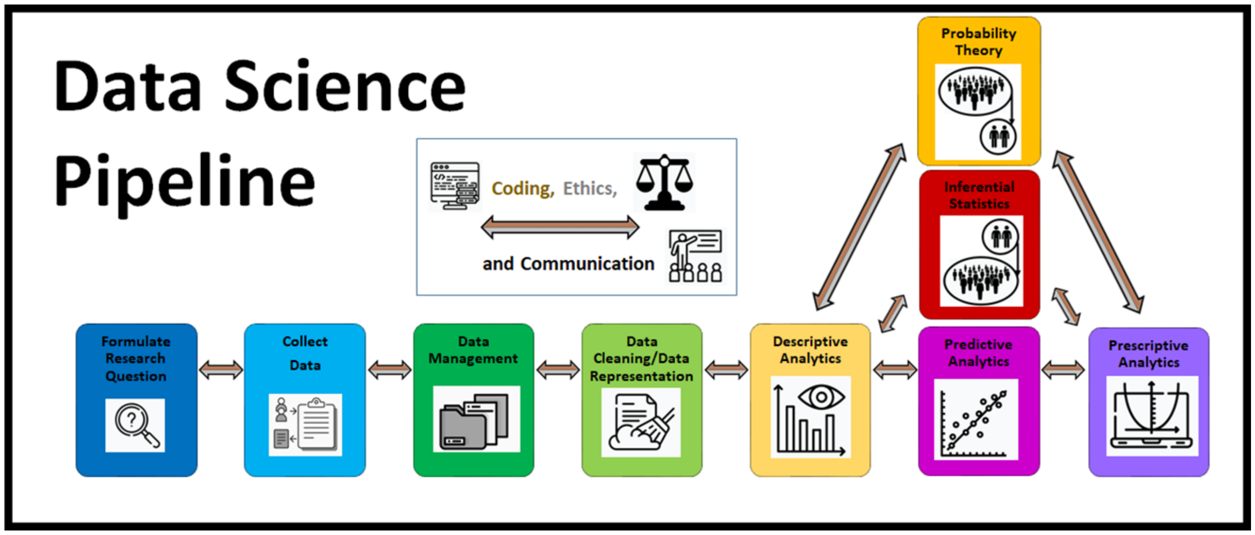


Figure 3: Graphic summary of the Data Science Pipeline. Figure by Ellison and Deeke [2].

2.2.3 Other methods

In semi-supervised learning, a model is trained on data related to one class, or some of the classes. Real life scenarios include anomaly detection where there is sufficient training data for normal events while anomalous ones occur rarely, expensive to acquire, or they evolve in a way that only deviations from the normal boundary can successfully detect them.

In self-supervised learning, features are learned using unlabelled data like unsupervised learning, however input-label pairs are constructed from each data point, enabling learning the structure of the data through supervised methods. Examples include word embeddings and autoencoders.

2.3 Representation Learning

Representation/feature learning is a broader term for techniques that allow an ML system to automatically discover representations needed for further tasks such as feature detection and classification. These tasks require inputs that are mathematically easy to represent and convenient to process and do operations with, however real world data is often the opposite (images, videos, sensor data, etc...). These representations are usually continuous vectors.

2.3.1 Autoencoders

An autoencoder is a type of neural network architecture that is having three core components: the encoder, the decoder, and the latent-space representation. The encoder compresses the input to a lower latent-space representation via weight matrixes biases and a nonlinear activation function and then the decoder reconstructs it.

2.3.2 Prototype Learning

Prototype learning is a family of methods where a model represents each class, cluster, or concept using prototypes—vectors in a learned feature space that act as representative examples. A model then makes predictions by comparing new samples to these prototypes. These prototypes provide an abstract representation for many natural categories and concepts.

2.4 K-means Clustering

Clustering is an unsupervised machine learning technique that learns patterns from the features of a dataset without using labels. *K-means* is a simple algorithm that attempts to partition samples into K groups with roughly equal variance, although the latter is not guaranteed.

K-means clustering is prototype learning algorithm used to categorize n (usually vectors of unlabelled data) items into k number of clusters. It is useful for identifying natural groupings of data and structuring raw data. Determining what k should be is important for the segmentation to be meaningful and there are multiple methods for determining the optimal value for k. K-means has a variety of uses due to its simplicity and effectiveness, including data segmentation, image compression and anomaly detection.

- Unsupervised machine learning technique
- "K-means" = "K averages" / "K central tendencies" / "K centroids"
- Can be looked at as a prototyping tool that discretizes the data into K prototypes

2.4.1 Algorithm in a nutshell

The algorithm will categorize the inputs into k clusters based on similarity. Measuring of similarity is done with the square of the Euclidean distance of the data vectors. Summarized, the algorithm works as follows:

1. Initialization: Randomly select k number of centerpoints for the clusters, called centroids.
2. Assignment: Each data point is assigned to a cluster based on which centroid is the nearest to it (using Euclidean distance).
3. Update: We recalculate the position of each centroid based on the average of the data points in each cluster.
4. Repeat: We repeat this process until the position of each centroid is unchanged or we reach a pre-defined iteration limit

2.4.2 Algorithm in details

According to the *Scikit Learn Documentation*, the K-means algorithm "divides a set of N samples X into K disjoint clusters C, each described by the mean μ_j of the samples in the cluster". These means are referred to as *centroids*, and despite living in the same space, they are usually not part of the dataset. The number K is required to be initialized, along with the starting positions of the centroids. The algorithm has an objective function that minimizes the criterion of *within-cluster sum-of-squares*, the so-called *inertia*.

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2)$$

Inertia is "a measure of how internally coherent clusters are". It assumes that clusters are convex and isotropic. As the documentation highlights it, it suffers from drawbacks:

- performs poorly on elongated clusters or manifolds with irregular shapes
- not normalized: lower values are better, zero is optimal, yet in higher-dimensions distances get inflated

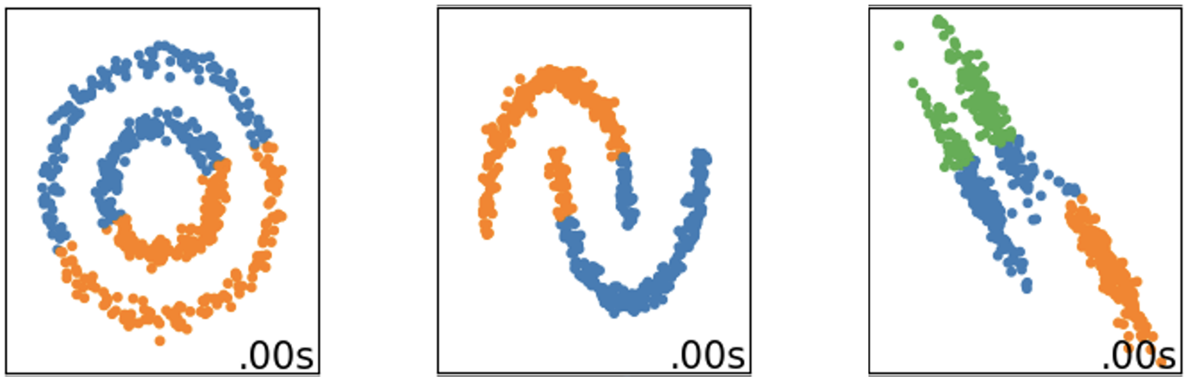


Figure 4: K-means behavior illustrated on simple synthetic datasets with different shapes.

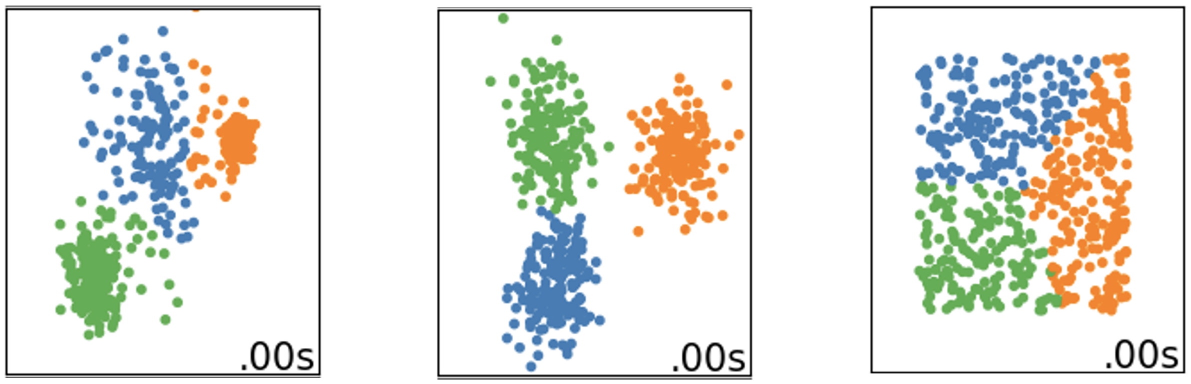


Figure 5: K-means behavior illustrated on simple synthetic datasets with different shapes.

The algorithm follows an iterative process that consists of three main steps.

- initialize centroid positions
- assign each sample to its nearest centroid
- create new centroids by taking the mean value of the samples assigned to each previous centroid,

then the difference between old and new centroids are calculated and the algorithm loops between steps 2) and 3) until this difference becomes small considering a threshold.

The documentation highlights yet another weakness of the algorithm at the stage of centroid initialization, as written, "K-means will always converge, however this may be to a local minimum". This outcome depends on the initial positions of the centroids, which if assigned randomly, may not find every cluster in the dataset. To address this, the algorithm in the *sklearn* library can be instructed to use a more intelligent initialization technique that places centroids distant from each other and can be used via the *init* = "k-means++" argument value. Besides the *n_init* parameter can be used to enhance this process, which is the "number of times the k-means algorithm is run with different centroid seeds" where the candidate with the best inertia is kept.

2.5 Neural Networks

In order to comprehend the function of deep learning in unsupervised tasks, it is necessary to first define a neural network in its broadest definition. Neural networks, which are built to imitate biological processes, are fundamentally the engines for modern artificial intelligence. A neural network is described by *Amazon Web Services (AWS)* as a "method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain."

2.5.1 Layered structure

According to *GeeksforGeeks*, Artificial Neural Network (ANN) layers can be explained as follows:

1. **Input Layer:** This is the network's entry point. It transfers raw data—such as an image's pixels or a dataset's columns—to the layers that follow. Key role: this layer merely buffers and sends the input signals to the buried levels; no calculation takes place here.
2. **Hidden Layers:** The real "learning" and processing takes place in these layers, which are situated between the input and output layers. A network is referred to as "Deep Learning" if

it has one or more hidden layers. Key role: they use mathematical processes (weighted sums) and activation functions (like *sigma* or *ReLU*) to add non-linearity to extract features and patterns from the data.

The Hidden Layer types can vary depending on the architecture:

- **Dense (Fully Connected) Layers:** Every neuron connects to every neuron in the next layer
- **Convolutional (Convergent) Layers:** Mostly used to identify geometric sequence in images
- **Recurrent (Repetitive) Layers:** Applied to sequence data, such as text or periods of time
- **Pooling & Dropout Layers:** used to avoid overfitting and minimize dimensions, respectively

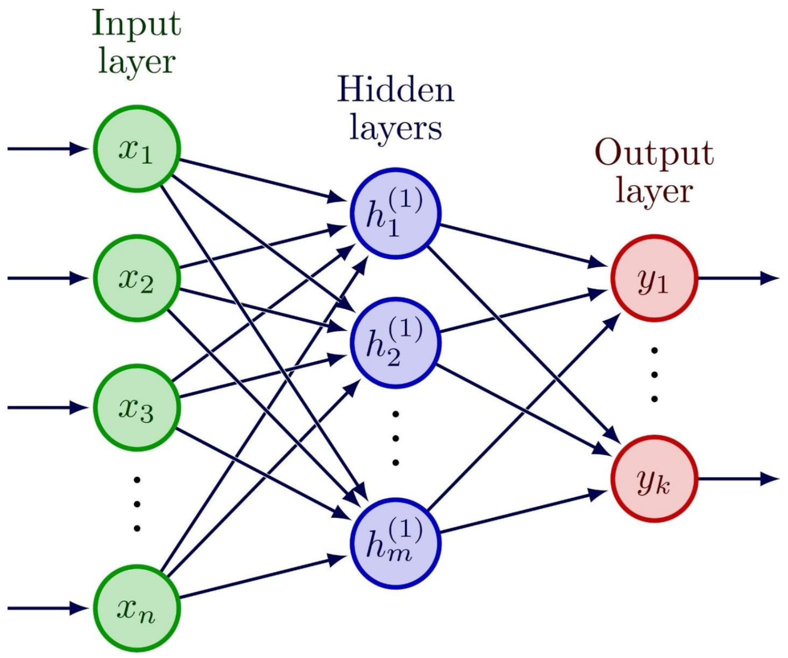


Figure 6: A diagram of a feedforward artificial neural network illustrating the flow of data through input, hidden, and output layers.

AWS states that "individual nodes can be simple, but when connected in a dense network, they solve complex problems." Supervised learning, in which the network is trained on labeled data (e.g., displaying computer images labeled "cat" until it learns to identify a cat), is traditionally the most well-known use of this design. The data isn't always labeled, though. This leads us to unsupervised techniques such as clustering.

2.5.2 Neural Networks in Clustering

Clustering is a key method in unsupervised learning, claims *IBM*. It involves "identifying distinct groups of data points" in which the machine must independently recognize patterns without the need for human assistance or pre-existing classifications. According to *IBM*, "clustering algorithms identify distinct groups of data points... such that data points in the same group are more similar to other data points in the same group than those in other groups."

A shift in architecture is necessary to combine these two concepts—by utilizing a Neural Network, which is typically supervised, with Clustering, that is unsupervised. How does the network learn if there are no labels to fix it? Standard feed-forward networks aren't ideally suited for this, as discussed in the *Cross Validated (Stack Exchange)* forum. Rather, certain designs are needed. The usage of automatic encoders is one popular strategy that was brought up in the conversation. One contributor points out that clustering can be accomplished by “training an autoencoder... and then clustering the data in the bottom layer.” In this case, the clustering process is far more effective than attempting to group raw, high-dimensional data since the neural network learns to compress data (dimensionality reduction) into a dense representation.

However, there is a more direct “neural” approach to clustering that relies on competitive learning instead of error correction which is *Self-Organizing Map (SOM).*

2.6 Self-Organizing Maps (SOM)

According to GeeksforGeeks [3], a Self-Organizing Map is an unsupervised neural network algorithm that maps high-dimensional data to a lower dimensional grid making interpretation and visualization of data easier. It follows both competitive and collaborative learning to fit a dataset using neighborhood defining parameters and functions to not only update the best matching unit of a given sample but also its neighborhood with a decay over a certain number of epochs. The learning process is performed in the following steps:

1. **Setup:** The grid of the SOM is defined by its dimensions and hyperparameters such as the neighborhood definitions.
2. **Initialization:** The weights (the positions in the N-dimensional space) of the grid nodes are initialized randomly.
3. **Competition:** For a given sample, the distance between each node in the grid is calculated and the closest is designated as the winning, that is the *best matching unit (BMU).*
4. **Collaboration:** The weight vector of the BMU is shifted towards the weight vector of the sample data using the learning rate parameter. The topological neighbors of the BMU are also updated according to the neighborhood radius (sigma) and the neighborhood function.
5. **Decay:** Over the epochs, the sigma and learning rate decay. In the beginning, larger updates are made to help organization of the grid, but later only smaller refinements are performed to finalize convergence to local patterns.
6. **Stopping:** The learning finishes when the maximum number of epochs is reached or when early stopping is instructed.

3 Methodology

1. Use *K-means* to cluster data points in their original N-dimensional space and visualize the cluster labeling in two-dimensional plots. Interpret the results, take notes of any strange patterns caused by the two-dimensional output, if there's any. Visually compare and evaluate the clustering against ground truth, and with the help of unsupervised metrics.
2. Use *Self-Organizing Maps (SOM)* to learn a representation of the data that can be visualized in two dimensions through the organized and flattened grid of the SOM.
3. The SOM representation itself discretized the dataset yet if non-linear patterns are to be captured a larger, more flexible grid is required, by which it is expected that a real cluster will covered by multiple nodes in the grid. Therefore, we can apply *Agglomerative Clustering* to the learned representation to find high-level clusters in it. Use these cluster labels obtained via the best-matching unit (BMU) in the representation for each data point. Visually compare and evaluate the clustering against ground truth, and with the help of unsupervised metrics.
4. Compare the results obtained via the algorithms of *K-means* and *Self-Organizing Maps (SOM)*.
5. Briefly introduce a better visualization of the flattened two-dimensional *Self-Organizing Maps (SOM)* visual.

4 Experiments

4.1 Dataset

The `Palmer penguins` dataset was used during the experiments obtained from the `palmerpenguins` python library. The original shape of the dataset is 344 observations and 8 features, both categorical and numerical.

4.2 Preprocessing

The dataset was limited to the 4 numerical features of `bill_length_mm`, `bill_depth_mm`, `flipper_length_mm` and `body_mass_g`. Observations with missing values in these features were dropped, resulting in a total of 342 records. The features of this dataset were standardized.

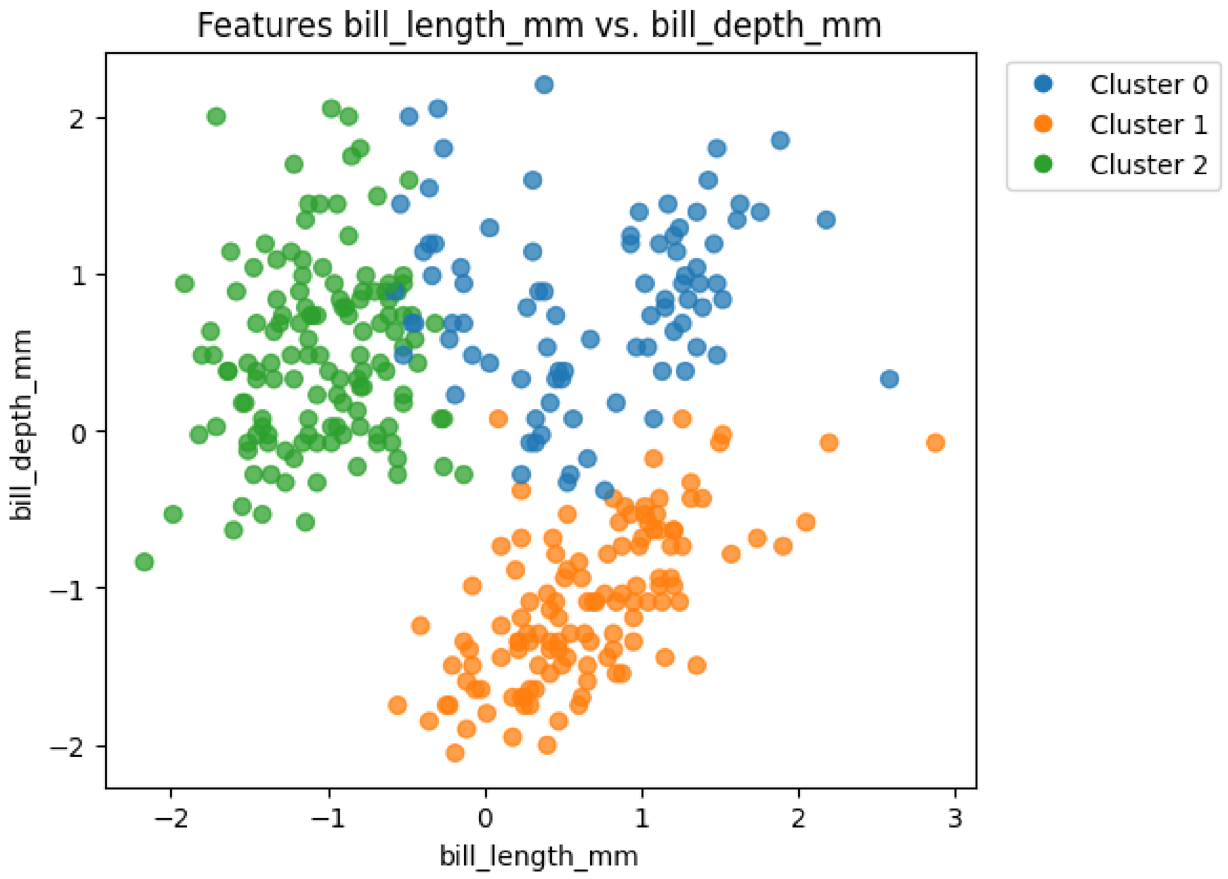
4.3 Clustering

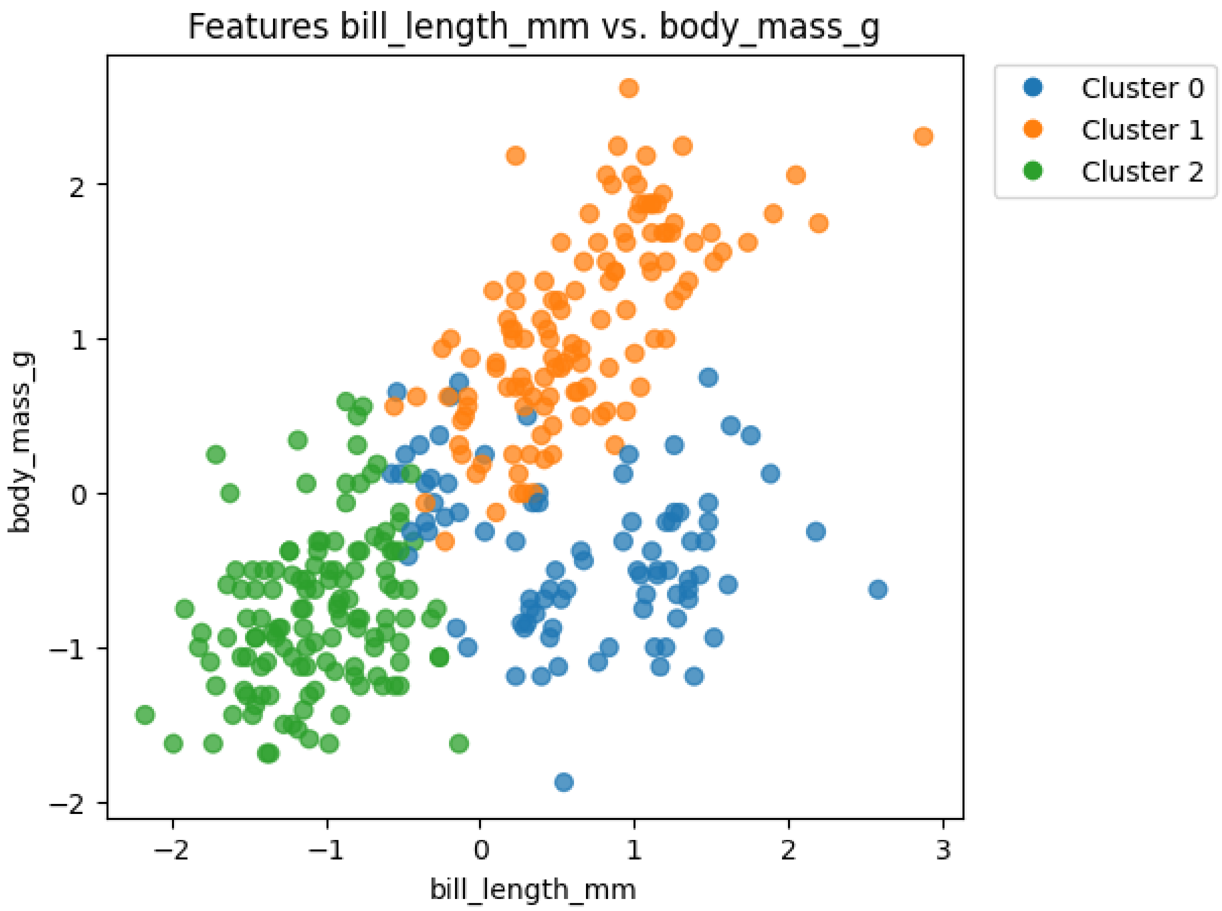
A `KMeans` estimator from the `sklearn` library was used to fit and cluster the data in its preprocessed four-dimensional space with a heuristic to separate 3 groups.

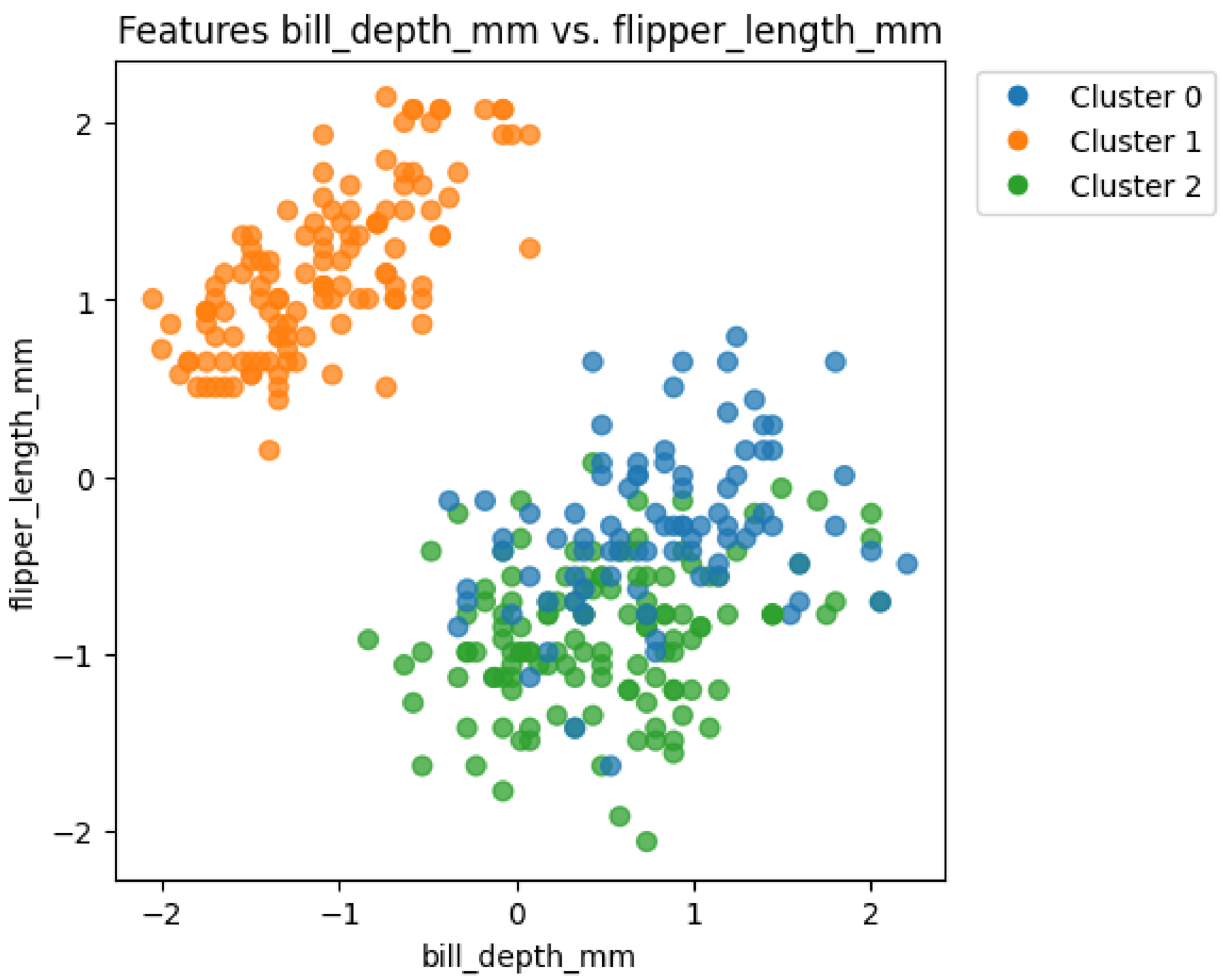
To interpret the results, the data labeled by cluster ids was plotted in two-dimensional scatter plots, for all features, pairwise. Despite only a few dimensions, this resulted in 6 individual plots. Examples are included below.

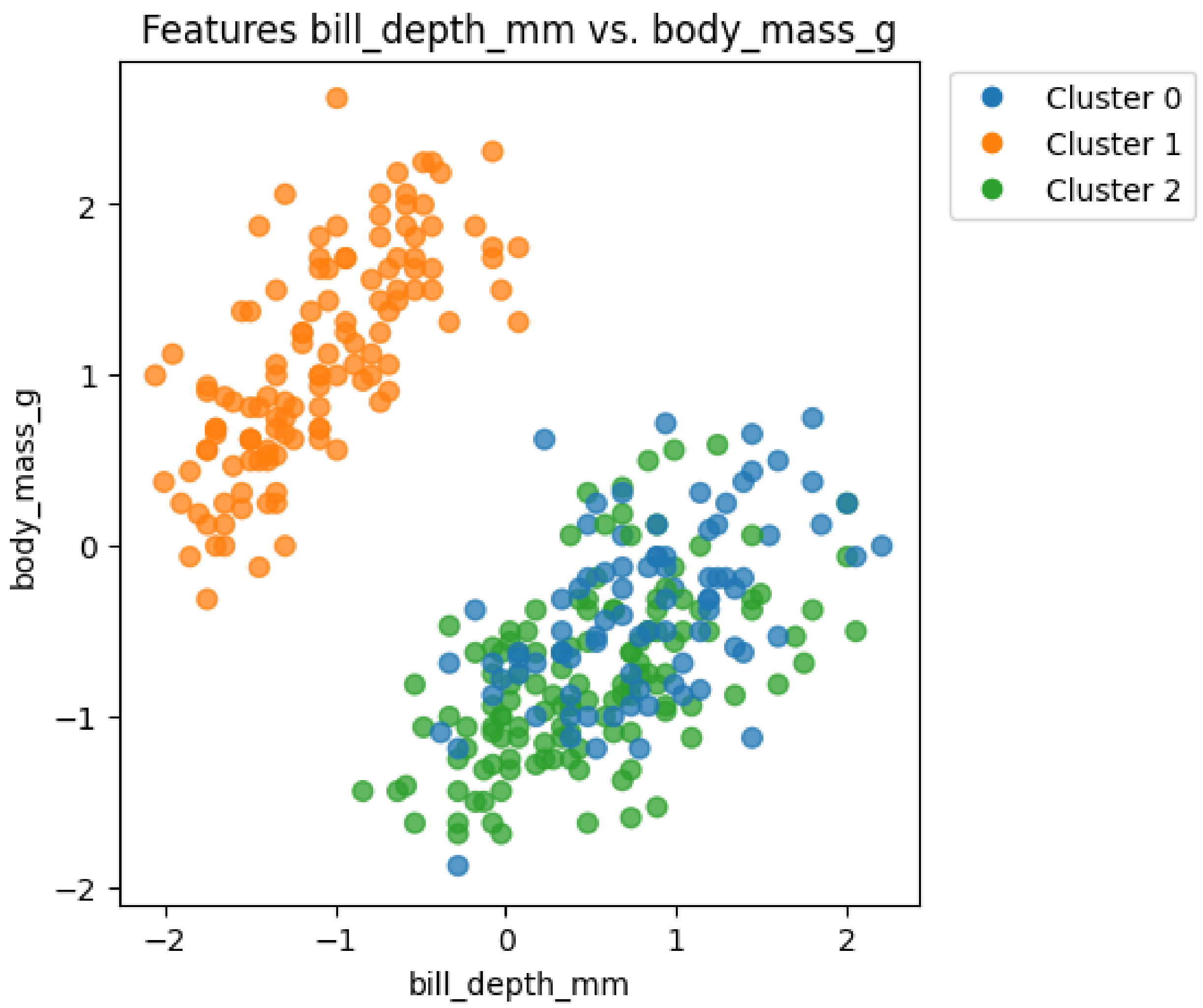
One important insight is that clusters do not separate clearly and there are regions where they overlap each other in the resulting plots, that may generate confusion in the observer. To resolve this, we list the reasons for this phenomenon:

- Dataset itself **may not have clear separation** lines for the real groups: Clusters meet which can be amplified by the styling properties of the scatter plot such as the size of the marker
- **Projection** from 4D to 2D **distorts** the **original structure**: In the original feature space those marked clusters may not overlap as they appear in the 2D collapsed view
- **K-Means assumptions**: K-Means divides the data space into convex spherical clusters, that may lead to arbitrary separation lines that do not align with irregularly shaped clusters


Features bill_length_mm vs. bill_depth_mm
bill_depth_mm
bill_length_mm
2
1
0
−1
−2
−2
−1
0
1
2
3
Cluster 0
Cluster 1
Cluster 2


Features bill_length_mm vs. body_mass_g
Cluster 0
Cluster 1
Cluster 2
body_mass_g
bill_length_mm
−2
−1
0
1
2
3


Features bill_depth_mm vs. flipper_length_mm
flipper_length_mm
bill_depth_mm
Cluster 0
Cluster 1
Cluster 2
2
1
0
−1
−2
−2
−1
0
1
2



References

[1] DataScience-PM.com. *What is a Data Science Workflow?* Accessed: 2025-12-26. 2024. URL: https://www.datascience-pm.com/data-science-workflow/.

[2] Tori Ellison and Julie Deeke. *Data Science Exploration. A second-semester exploration into the world of data science*. Accessed: 2025-12-26. 2025. URL: https://exploration.stat.illinois.edu/.

[3] GeeksforGeeks. *Self Organizing Maps – Kohonen Maps*. Last updated June 26, 2025; Accessed: 2025-12-26. 2025. URL: https://www.geeksforgeeks.org/python/self-organising-maps-kohonen-maps/.