Master in Statistics and Operations Research

Optimization in Data Science

Clustering

Víctor Diví i Cuesta

Ivan Parreño Benítez

Universitat Politècnica de Catalunya

Facultat de Matemàtiques i Estadística

Contents

1. Introduction 1
2. Dataset 1
3. Clustering 3
   3.1. Exact K-Medoids 3
   3.2. Minimum Spanning Tree 4
   3.3. K-Means 7
   3.4. K-Medoids 8
4. Comparison 8
Bibliography 10

List of Figures

Figure 1 The Palmer Archipelago penguins. Artwork by @allison_horst. 1
Figure 2 Correlation matrix of dataset features. 2
Figure 3 K-Medoids AMPL clustering results 4
Figure 4 K-Medoids AMPL confusion matrices 4
Figure 5 Clustering library example 5
Figure 6 MST clustering results 6
Figure 7 MST confusion matrices 6
Figure 8 MST-H clustering results 6
Figure 9 MST-H confusion matrices 7
Figure 10 K-Means clustering results 7
Figure 11 K-Means confusion matrices 8
Figure 12 K-Medoids clustering results 8
Figure 13 K-Medoids confusion matrices 8
Figure 14 Results of computational times for each clustering method. 9

List of Tables

Table 1 Features of the Palmer Penguins Dataset 1
Table 2 ANOVA F-Values and related P-Values of each feature. 2
Table 3 Feature weights for the MultiVariate Scaled scenario. 3
Table 4 Classification results for every clustering method and scenario by species. 9

List of Codes

Code 1 AMPL Model for the K-Medoids. 3
Code 2 MST Clustering using the `mst_clustering` library. 5

1. Introduction

In this project, several clustering methods are explored and compared using the Palmer Penguins Dataset (as both a multi and bivariate dataset). The project is structured as follows: Section 2 presents and explains the used dataset, Section 3 presents the methods used and their results, and Section 4 shows a comparison of the results obtained with every method.

2. Dataset

The Palmer Penguins dataset [1] is an introductory dataset widely used in data science as an alternative to the well-known "Iris" dataset. The dataset is composed of data about three penguin species observed in the Palmer Archipelago, Antarctica.

In this project, we use a curated dataset that is provided in the `palmerpenguins` R[1] and Python[2] packages. In particular, it includes morphological and demographic measurements of 344 individual penguins, collected between 2007 and 2009 in the Palmer Long-Term Ecological Research[3], from three different species:

- Adélie (*Pygoscelis adeliae*)
- Chinstrap (*Pygoscelis antarcticus*)
- Gentoo (*Pygoscelis papua*)

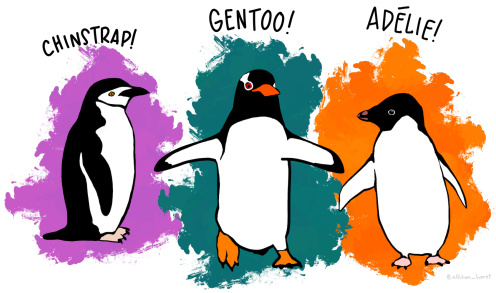


Figure 1: The Palmer Archipelago penguins. Artwork by @allison_horst.

The curated dataset contains the following features (some names differ from the raw data):

Feature	Description	Units/Values
`species`	Penguin species	`Adelie, Chinstrap, Gentoo`
`island`	Island in the Palmer Archipelago	`Biscoe, Dream, Torgersen`
`bill_length_mm`	Length of the penguin's bill	millimeters
`bill_depth_mm`	Depth of the penguin's bill	millimeters
`flipper_length_mm`	Length of the penguin's flipper	millimeters
`body_mass_g`	Penguin body mass	grams
`sex`	Biological sex of the penguin	`male, female`
`year`	Year of the observation	`2007, 2008, 2009`

Table 1: Features of the Palmer Penguins Dataset

Some preprocessing has been performed on the dataset before applying the clustering algorithms:

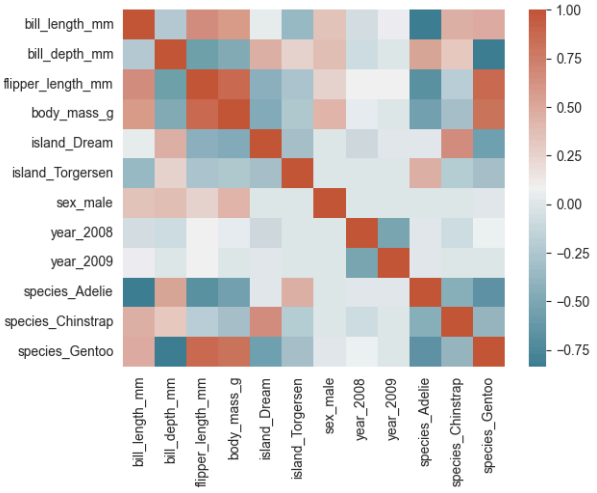
- **Data cleaning:** 11 measurements that contained null values in at least one feature have been removed.

[1] https://allisonhorst.github.io/palmerpenguins/articles/intro.html
[2] https://github.com/mcnakhaee/palmerpenguins
[3] https://pallter.marine.rutgers.edu/

- **Categorical encoding**: categorical features (`island`, `sex`, and `year`) have been transformed into numerical ones using a dummy encoding, increasing the number of features from 8 to 10.
- **Normalization**: all features have been normalized applying a Standard Score (or Z-Score) Normalization, since it is not affected by outliers in the dataset.

To decide which features to use in both the multivariate and the bivariate clustering, we have analyzed the correlation between all the features and performed an Analysis of Variance (ANOVA) to analyze the degree of linear dependency between the different features and the species.

Figure 2 shows the correlation matrix as a heatmap, where colors close to white indicate no correlation, while dark red and blue indicate strong correlation (positive and negative respectively). As can be seen, many of the features are significantly correlated (e.g. `flipper_length_mm` with `bill_depth_mm` and `body_mass`), and some features are also heavily correlated with the species (e.g. `bill_length_mm` with the Adélie species and `flipper_length_mm` with both Adélie and Gentoo species). We can also see that some features have almost no correlation with the species, namely the sex of the penguins and the year of the measurement, although they do have some correlation with the other features. These insights are further supported by the ANOVA results shown in Table 2, which presents the F-Statistic for each feature and the related P-Value (higher F-Statistic indicate significant differences of the feature among species and the P-Value indicate the probability of this differences being by chance).



Feature	F-Statistic	P-Value
flipper_length_mm	567.406992	1.587418e-107
bill_length_mm	397.299437	1.380984e-088
bill_depth_mm	344.825082	1.446616e-081
body_mass_g	341.894895	3.744505e-081
island_Dream	208.347193	3.063542e-059
island_Torgersen	43.988989	1.160083e-017
year_2008	1.245831	2.890514e-001
sex_male	0.024088	9.762014e-001
year_2009	0.019740	9.804544e-001

Table 2: ANOVA F-Values and related P-Values of each feature.

Figure 2: Correlation matrix of dataset features.

For the bivariate dataset, we have selected `flipper_length_mm` and `bill_length_mm`, which are the most significant features according to the ANOVA.

Due to the low relevance of the `sex` and `year` features, they have been discarded from the multivariate dataset. Although some extra treatment of the dataset could be perform, such as a Principal Component Analysis (PCA) to reduce dimensionality and minimize correlation between features, we decided against it in favor of having features with real-life meaning.

3. Clustering

In this section, multiple clustering methods are explored, both exact and heuristic. For each method, three different clusterings have been performed: MultiVariate, MultiVariate Scaled, and BiVariate.

The MultiVariate and BiVariate scenarios use the normalized dataset with the 6 and 2 selected features respectively. However, since we are using an euclidean distance for all the methods, we are giving the same weight to every dimension, even though we showed in the previous section that some features are significantly more relevant than others. Taking this into account, and to explore how a different distance affects the clusterings, we introduce the MultiVariate Scaled scenario, which adds "weights" to the most important features so they become more relevant in the clustering. To do so, we simply scale the features according to their weight, so that the resulting euclidean distance is also multiplied. The weights used are shown in Table 3.

Feature	Multiplying Factor
flipper_length_mm	3
bill_length_mm	2
bill_depth_mm	1.75
body_mass_g	1.75

Table 3: Feature weights for the MultiVariate Scaled scenario. Other features have weight of 1.

All functions using the different clustering methods are provided in independent files named accordingly to the method they use.

3.1. Exact K-Medoids

```
# Set of points
set I;                          # Total points

# Parameters
param d {i in I, j in I};       # Distance from point i to medoid j
param k > 0;                    # Number of clusters

# Variables
var x{i in I, j in I} binary;# 1 if point i is assigned to medoid j, else 0

# Objective function: minimize the sum of distances
minimize clustering:
    sum {i in I, j in I} x[i,j] * d[i,j];

# Constraints

# Each point must be assigned to exactly one medoid
subject to OneAssignment {i in I}:
    sum {j in I} x[i,j] = 1;

# Exactly k medoids
subject to Kclusters:
    sum {j in I} x[j,j] = k;

# A point can only be assigned to an active medoid
subject to ClusterExists {i in I, j in I}:
    x[i,j] <= x[j,j];
```

Code 1: AMPL Model for the K-Medoids.

The K-Medoids formulation problem was implemented in AMPL, using the formulation presented in class, where every point has a distance to every other point in the clustering.

To represent the assignments, a matrix of size n × n is used, where each entry indicates whether a point is assigned to a specific medoid.

The constraints are as follows:

- **OneAssignment:** Each point must be assigned to one and only one medoid.
- **KClusters:** Only k points can be medoids. In the x matrix, this means that only k diagonal elements (where i = j) take the value 1.
- **ClusterExists:** A point can only be assigned to a cluster if that cluster (medoid) actually exists.

Finally, the objective function minimizes the total distance between each point and its assigned medoid.

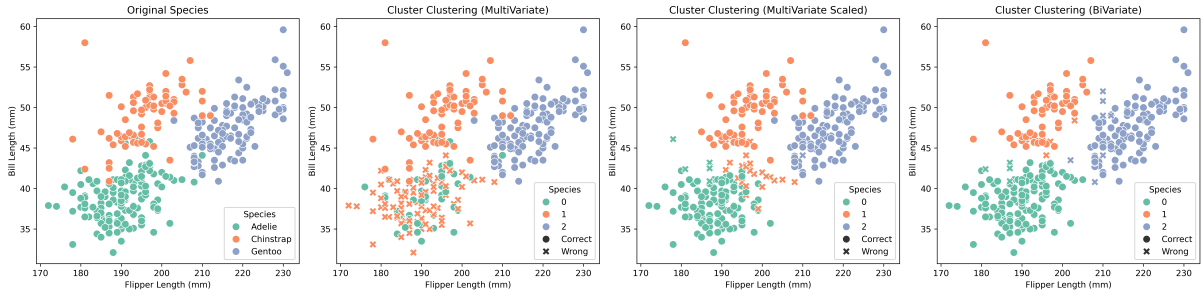


Figure 3: K-Medoids using AMPL clustering results comparing from left to right, real clusters, MultiVariate results, MultiVariate Scaled results and BiVariate results.

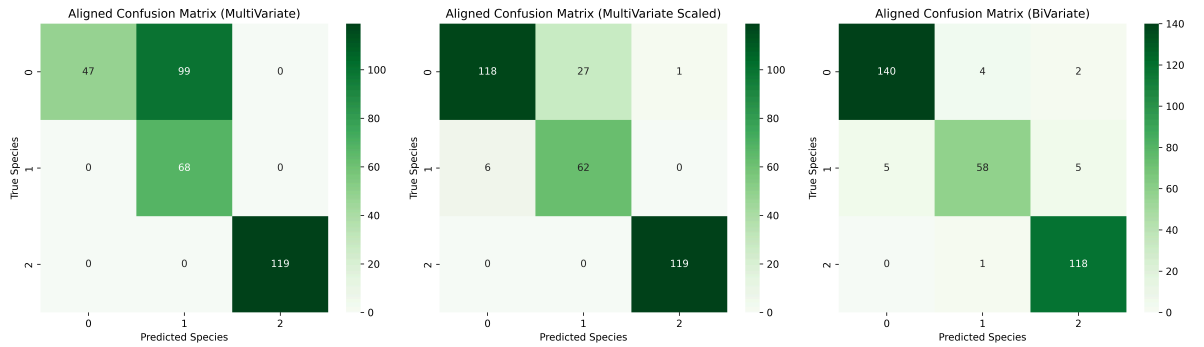


Figure 4: K-Medoids using AMPL clustering confusion matrices of MultiVariate results (left), MultiVariate Scaled results (middle), and BiVariate results (right).

3.2. Minimum Spanning Tree

The function for generating a Minimum Spanning Tree (MST) is shown in Code 2, and uses the `mst_clustering` library, which automatically builds the MST from the data [2]. This library allows us to adjust several parameters to control the clustering process:

- **cutoff**: determines how many edges in the MST are removed. This results in the creation of cutoff + 1 clusters.
- **metric**: specifies whether the input data already represents a distance matrix or whether the library should compute the distances itself.

- **approximate**: if set to TRUE , the algorithm computes an approximate MST instead of an exact one. This can speed up the process for large datasets but may slightly affect precision, so we decided to compute the best possible MST.

```python
import pandas as pd
from mst_clustering import MSTClustering

def run_mst_clustering(data: pd.DataFrame, cutoff: int, approximate=False):
    model = MSTClustering(
        cutoff=cutoff,
        metric="euclidean",
        approximate=approximate,
    )

    labels = model.fit_predict(data.values)

    return labels

```

Code 2: MST Clustering using the `mst_clustering` library.

Figure 5 shows an example in which it is quite easy to see which edges will be cut if we create an MST, given that the 4 clusters are well isolated.

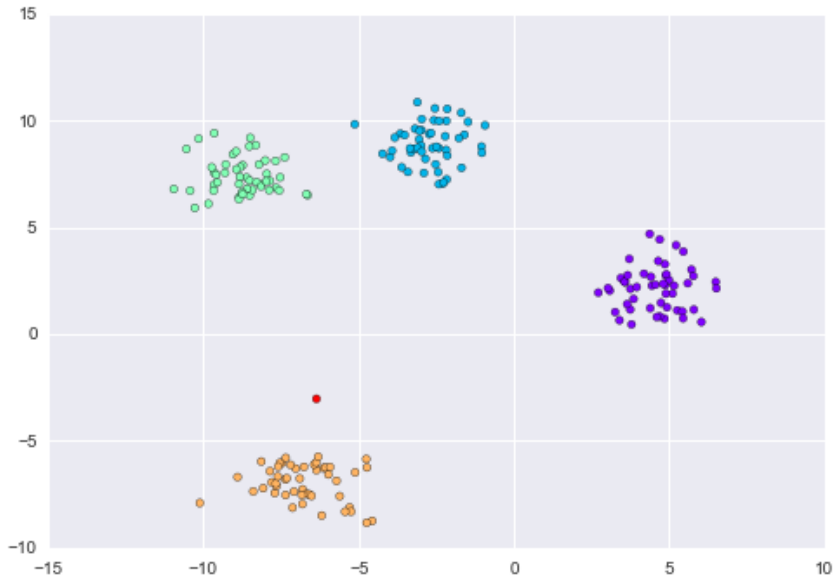


Figure 5: Example of a clustering with four groups where using k+1 as cut yields the same number of clusters.

As shown in the MST plot (Figure 6), the results are not very promising. In the bivariate case, the method generates three clusters—two containing a single point and one large cluster with the rest. For the scaled data, it detects only one point from the Chinstrap group and forms two large clusters with the remaining points. In the multivariate case, the method struggles with the Adelie group, failing to separate it properly. This occurs because outliers create large distance gaps, leading to poor cluster separation overall.

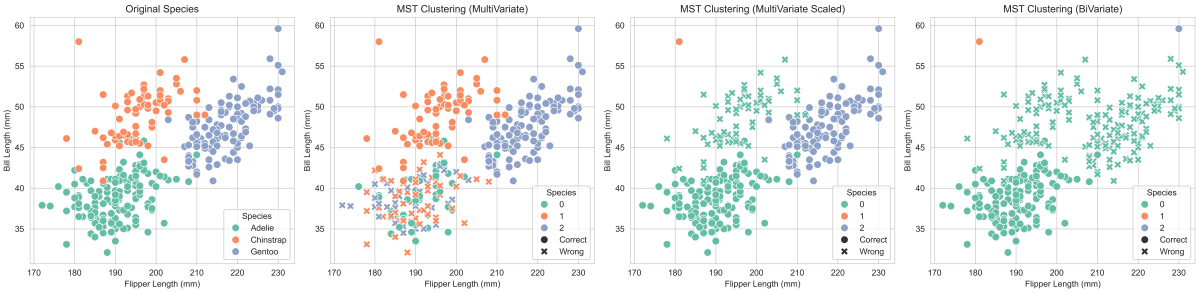


Figure 6: MST clustering results comparing, from left to right, real clusters, MultiVariate results, MultiVariate Scaled results and BiVariate results.

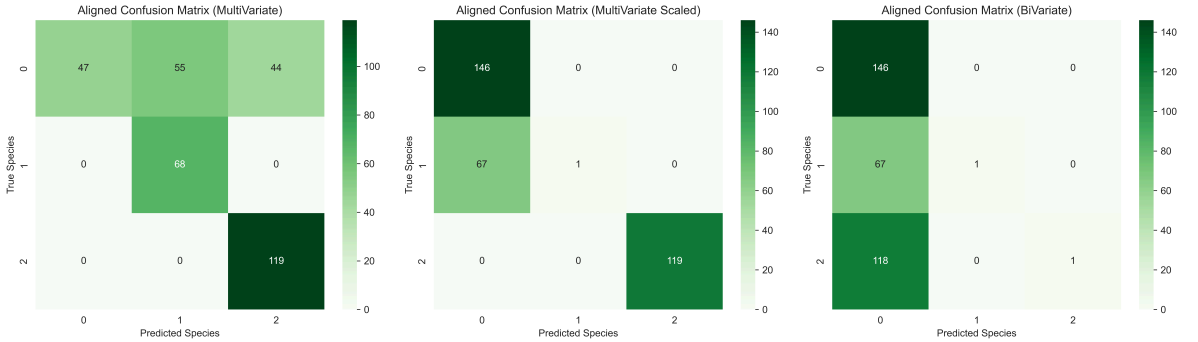


Figure 7: MST clustering confusion matrices of MultiVariate results (left), MultiVariate Scaled results (middle), and BiVariate results (right).

In order to obtain better results, we implemented a heuristic that treats clusters with only one point as outliers. We keep applying this process until we obtain cutoff + 1 clusters, each with a minimum size of 10. However, as can be observed, the results are still not very promising. The minimum size threshold helps to merge small but valid clusters, but the MST structure is sensitive to local density variations. This heurisitic also considers that points that do not belong to any cluster (-1 in the plots), are considered outliers and noise.

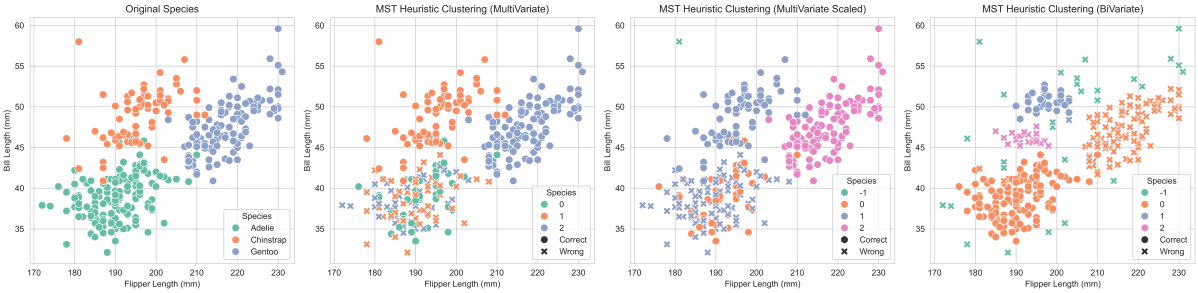


Figure 8: MST clustering heuristic results comparing, from left to right, real clusters, Multi-Variate results, MultiVariate Scaled results and BiVariate results.

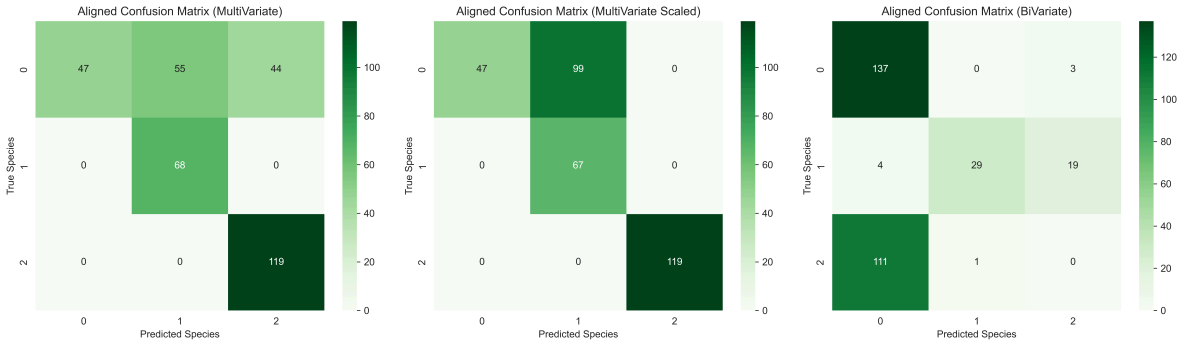


Figure 9: MST clustering heuristic confusion matrices of MultiVariate results (left), MultiVariate Scaled results (middle), and BiVariate results (right).

3.3. K-Means

The K-Means clustering has been performed using the `scikit-learn` library using the default initialization, the Greedy K-Means++ [3], which samples the dataset using the probability distribution of the points' contribution to the overall inertia[4].

Figure 10 shows the results of the K-Means clustering for the three scenarios, while Figure 11 shows their confusion matrices. As can be clearly seen, the MultiVariate scenario groups all Gentoo penguins perfectly, with neither false positives nor false negatives, and classifies all Chinstrap correctly, but misclassifies more than one third of the Adélie penguins as Chinstrap.

Nothing of this happens in the Scaled or the BiVariate scenarios, in which every cluster contains the majority of the penguins of the corresponding species, but certain individuals from all three species are misclassified.

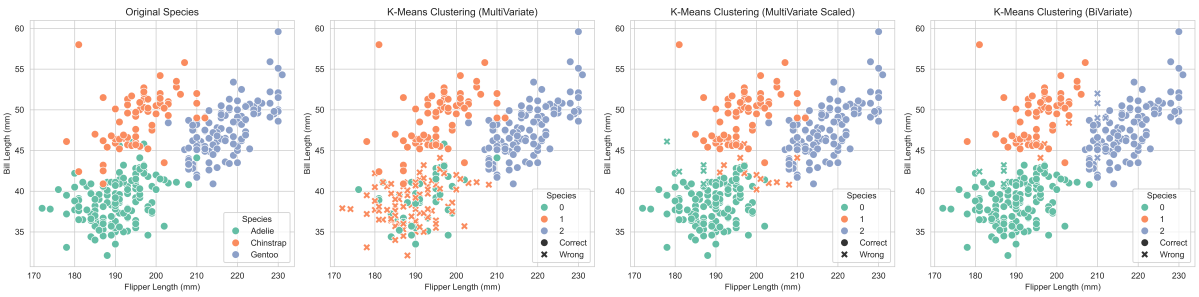


Figure 10: K-Means clustering results comparing, from left to right, real clusters, MultiVariate results, MultiVariate Scaled results and BiVariate results.

[4]Extracted from scikit-learn's documentation: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

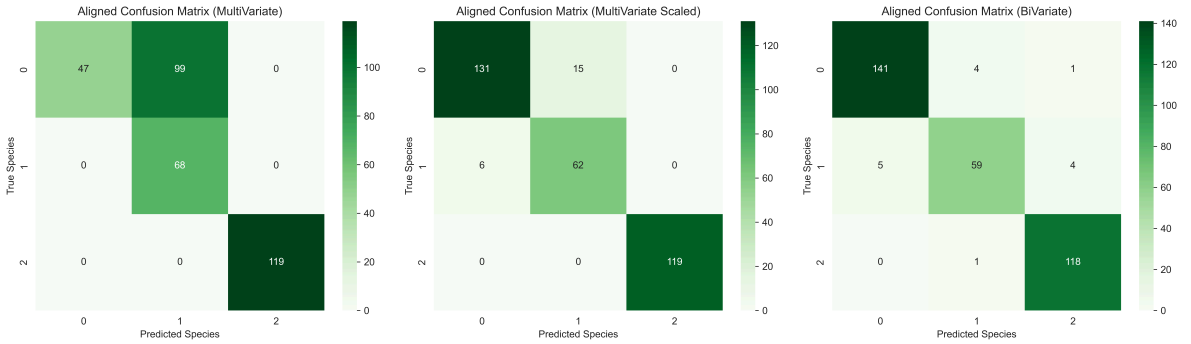


Figure 11: K-Means clustering confusion matrices of MultiVariate results (left), MultiVariate Scaled results (middle), and BiVariate results (right).

3.4. K-Medoids

The K-Medoids clustering has been computed using the `kmedoids` Python library using the default configuration, which uses the FasterPAM algorithm [4].

Figure 12 shows the results of the K-Means clustering for the three scenarios, while Figure 13 shows their confusion matrices. The obtained results are almost identical to the ones obtained with the K-Means clustering in Section 3.3. In the MultiVariate scenario, the results are exactly the same, while in the two others the results are slightly worse (7 and 2 more misclassified penguins respectively).

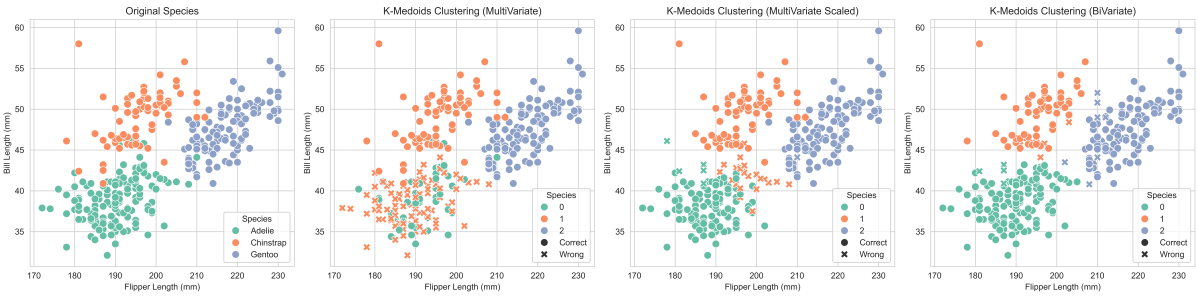


Figure 12: K-Medoids clustering results comparing, from left to right, real clusters, MultiVariate results, MultiVariate Scaled results and BiVariate results.

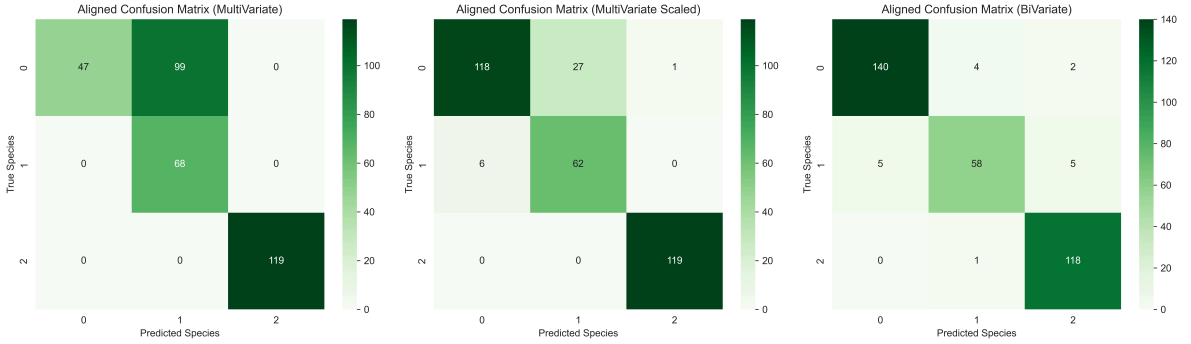


Figure 13: K-Medoids clustering confusion matrices of MultiVariate results (left), MultiVariate Scaled results (middle), and BiVariate results (right).

4. Comparison

In this section, we present a comparison of the speed and results of the different clustering methods. A summary of the results across all clustering methods and scenarios segregated by

species is shown in Table 4. The total amount of correctly classified and misclassified penguins is also shown.

First, it is evident that the heuristic MST method performs poorly in terms of clustering quality. On the other hand, the heuristic versions of K-Means and K-Medoids give comparable results, which shows that their approximations are reasonable. The exact K-Medoids algorithm produces almost the same results as its heuristic counterpart, so it makes sense to look at computational times to decide which approach is better.

Clustering Method	**Scenario**	**Adélie**			**Chinstrap**			**Gentoo**			**Total**	
		H	FP	FN	H	FP	FN	H	FP	FN	H	M
K-Medoids Exact	MV	47	0	99	68	99	0	119	0	0	234	99
	MVS	118	6	28	62	27	6	119	1	0	299	34
	BV	140	5	6	58	5	10	118	7	1	316	17
K-Means	MV	47	0	99	68	99	0	119	0	0	234	99
	MVS	131	6	15	62	15	6	119	0	0	312	21
	BV	141	5	5	59	5	9	118	5	1	318	15
K-Medoids	MV	47	0	99	68	99	0	119	0	0	234	99
	MVS	118	6	28	62	27	6	119	1	0	299	34
	BV	140	5	6	58	5	10	118	7	1	316	17
MST	MV	47	0	99	68	55	0	119	44	0	234	99
	MVS	146	67	0	1	0	67	119	0	0	266	67
	BV	146	185	0	1	0	67	1	0	118	148	185
MST-H	MV	47	0	99	68	55	0	119	44	0	234	99
	MVS	47	0	99	67	99	0	119	0	0	233	99
	BV	137	115	3	29	1	23	0	22	112	166	138

Table 4: Classification results for every clustering method and scenario by species. Scenarios: MV (MultiVariate), MVS (MultiVariate Scaled), and BV (BiVariate). Columns: H (Hits), M (Misses), FP (False Positives), FN (False Negatives).

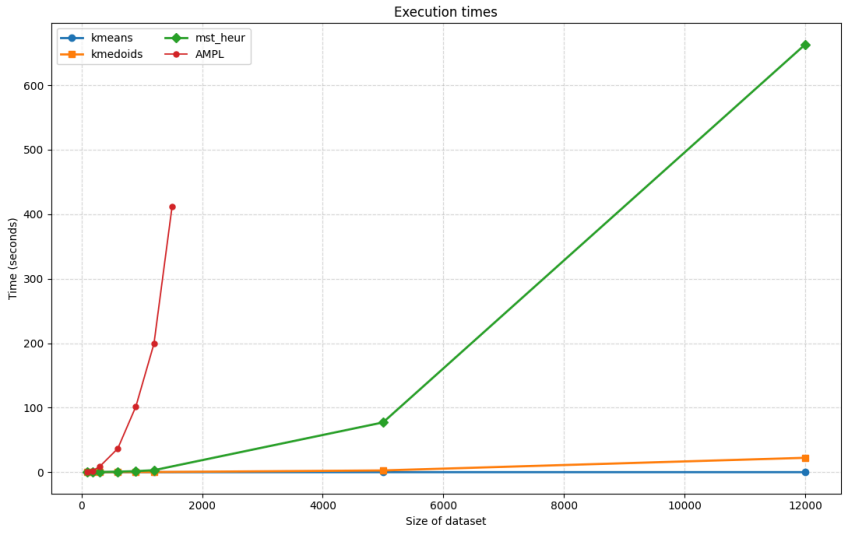


Figure 14: Results of computational times for each clustering method.

Figure 14 shows the execution time of the different clustering methods for datasets of different size. These datasets have been generated automatically based on the Palmer Penguins Dataset, keeping the distribution of each feature (although not the correlations).

As we can see in Figure 14, the computational time for exact K-Medoids grows exponentially. When n gets larger, the running times become very high, showing the scalability problems of the exact method. In practice, this makes heuristic methods or K-Means much more practical. K-Means has linear computational cost, and the heuristic K-Medoids implementation saves a lot of time, behaving almost like a linear method in practice.

Regarding MST methods, MST-H is clearly the slowest compared to the heuristics, and even if it can sometimes be faster than exact K-Medoids, its combination of clustering quality and computational cost makes it the least favorable choice overall. It is important to note that Normal MST is excluded from the comparison, as its results were not reliable enough to be considered a valid method. Only the multivariate version produced somewhat acceptable results, but they were far from sufficient. Given this, we can conclude that both Normal MST and MST-H represent the worst options for clustering.

Among all the methods explored, K-Means stands out as the best, providing a good balance between clustering quality and computational efficiency. It consistently gives accurate clusters while staying very fast, even as the dataset grows. For K-Medoids, the heuristic version is also a reasonable choice, since it seems to be as good as the exact method but can handle much larger datasets.

Finally, is important to note that an important factor affecting clustering performance is data preprocessing and variable weighting. When the dataset is balanced and more importance is given to the most relevant variables, all clustering methods improve significantly. This happens because:

- **Weighting important variables**: Amplifies meaningful differences between data points. Variables with high relevance contribute more to the distance or similarity metrics that drive clustering.
- **Noise reduction**: Giving less importance to irrelevant or noisy variables reduces their influence, preventing the algorithm from being "distracted" by dimensions that do not provide useful information.

This also explains why in K-Medoids and K-Means using bivariate or the scaled version we get better results than the multivariate one, as they focus on the most relevant features of the dataset.

Bibliography

[1] A. M. Horst, A. P. Hill, and K. B. Gorman, "palmerpenguins: Palmer Archipelago (Antarctica) penguin data." 2020. doi: 10.5281/zenodo.3960218.

[2] J. VanderPlas, "mst_clustering: Clustering via Euclidean Minimum Spanning Trees," *Journal of Open Source Software*, vol. 1, no. 1, p. 12, 2016, doi: 10.21105/joss.00012.

[3] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, in SODA '07. New Orleans, Louisiana: Society for Industrial, Applied Mathematics, 2007, pp. 1027–1035.

[4] E. Schubert and P. J. Rousseeuw, "Fast and eager k-medoids clustering: O(k) runtime improvement of the PAM, CLARA, and CLARANS algorithms," *Information Systems*, vol. 101, p. 101804, 2021, doi: https://doi.org/10.1016/j.is.2021.101804.