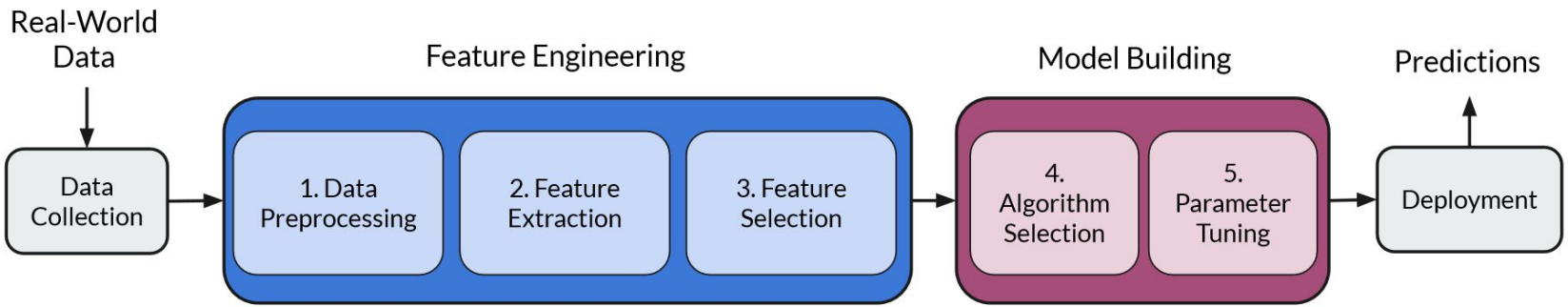
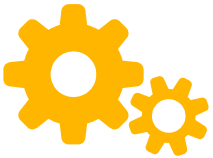
autocluster -

AutoML toolkit for automated clustering

PyPi: https://pypi.org/project/autocluster/
Github: https://github.com/wywongbd/autocluster/
Members: Wen Yan, Jung Chan, Yungi Jeong, Seungjun Lee

The ML pipeline





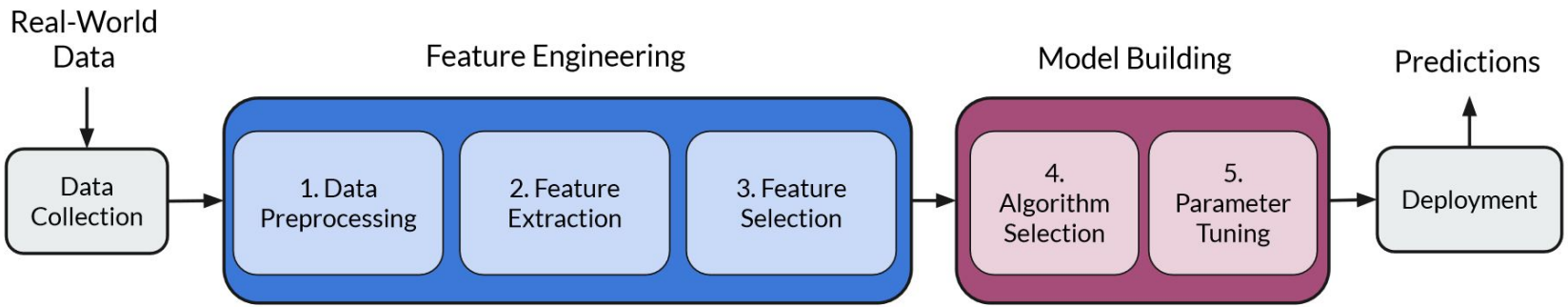
Every single pipeline stage is very difficult!

Each of these stages is time consuming, and requires experienced data scientists.

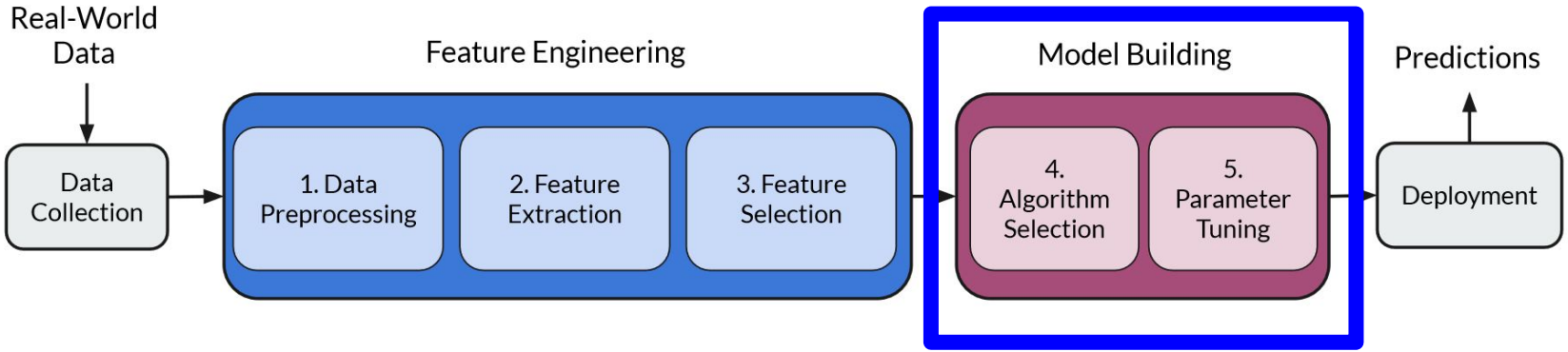
Why don't we *automate* it?

The goal of AutoML

To automate each stage of the pipeline!



Focus of our project



Combined Algorithm Selection and Hyperparameter optimization;

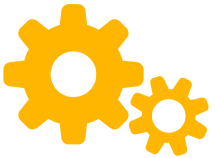
or

CASH problem

Formal definition of CASH

$$A^*, \lambda^* \in \underset{A^{(j)} \in \mathcal{A}, \lambda \in \Lambda^{(j)}}{\operatorname{argmin}} \frac{1}{K} \sum_{n=1}^{K} \mathcal{L}(A^{(j)}, \lambda, {D_{train}}^{(i)}, {D_{valid}}^{(i)})$$

A^*, λ^* are the optimal configurations from the search space which minimizes the k-fold cross validation loss.



Existing AutoML packages

Auto-sklearn

Automated model selection and hyperparameter optimization for regression and classification models in scikit-learn.

Featuretools

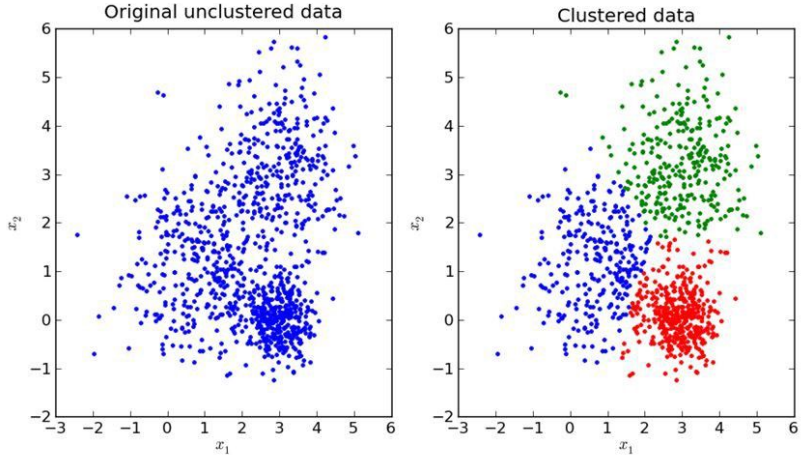
Performs automated feature engineering.

What about unsupervised learning tasks, like clustering?

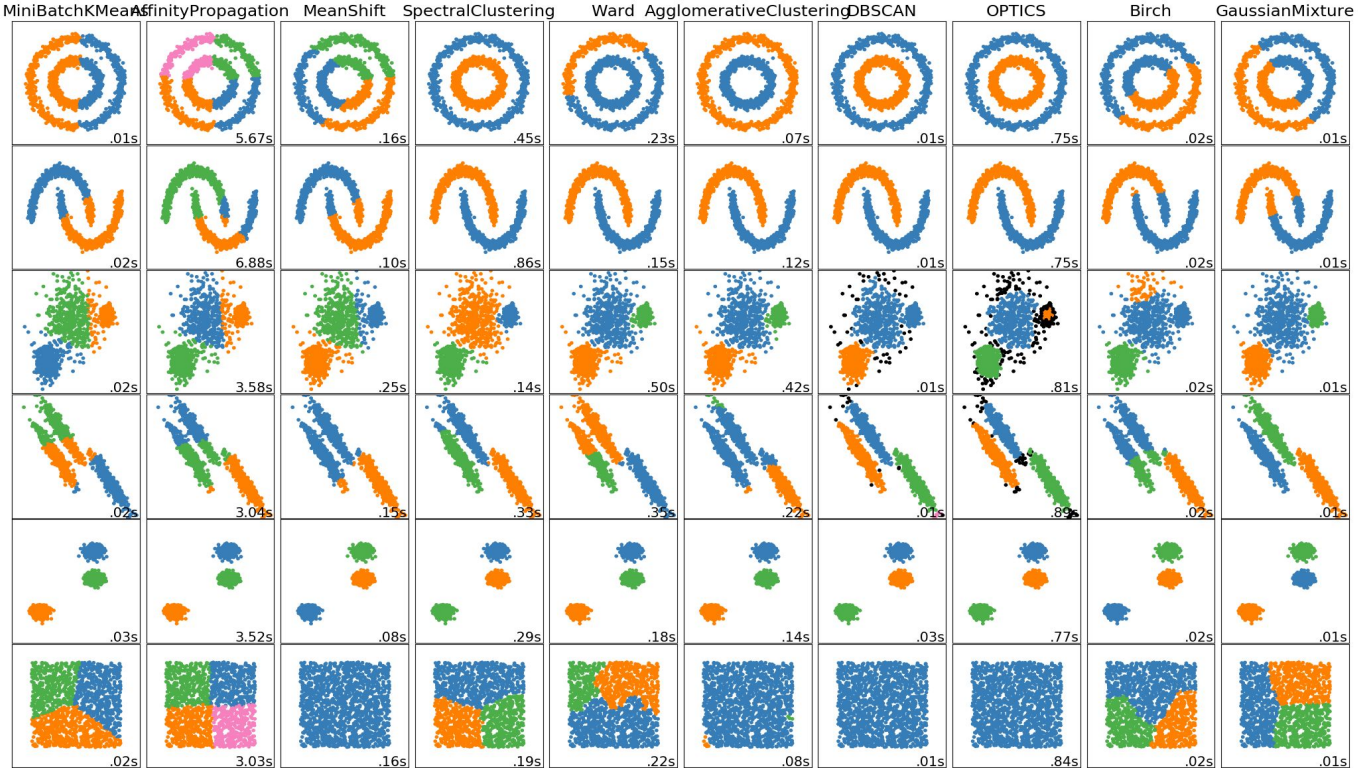
Supervised vs. Unsupervised

Datasets are labeled.

Datasets are unlabeled.



Clustering Algorithms in scikit-learn



Project Scope and objectives:

Areas of focus	Clustering	CASH Optimization	Meta-learning
Objectives	• Adopt existing blackbox optimization methods to solve CASH optimization problem for clustering tasks • Incorporate Meta-learning to enhance the convergence rate of CASH optimization • Develop a Python package for automated clustering		

1

Design & Implementation

Clustering, CASH Optimization, Meta-learning

CASH optimization for clustering

$$A^*, \lambda^* \in \underset{A^{(j)} \in \mathcal{A}, \lambda \in \Lambda^{(j)}}{\operatorname{argmin}} \frac{1}{K} \sum_{n=1}^{K} \mathcal{L}(A^{(j)}, \lambda, {D_{train}}^{(i)}, {D_{valid}}^{(i)})$$

The goal of CASH optimization is to identify the **optimal configuration** from the **search space** which minimizes the **objective function (k-fold cross validation loss).**

For a clustering task,

1 configuration = 1 choice of dimension reduction algorithm + 1 choice of clustering algorithm + 1 setting of relevant hyperparameters

CASH: Search space - dimension reduction

Method	Hyperparameters
TSNE	`n_components` (integer) `perplexity` (float) `early_exaggeration` (float)
PCA	`n_components` (integer) `svd_solver` (categorical) `whiten` (categorical)
Incremental PCA	`n_components` (integer) `whiten` (categorical) `batch_size` (integer)
Kernel PCA	`n_components` (integer) `kernel` (categorical)
Fast ICA	`n_components` (integer) `algorithm` (categorical) `fun` (categorical) `whiten` (categorical)

and more ….

7 dimension reduction algorithms in total.

In actual implementation, we made dimension reduction an optional procedure.

CASH: Search space - clustering algorithm

Method	Scalabillity	Hyperparameters
K-Means [2]	Not scalable	`n_clusters` (integer)
Mini Batch K-Means [34]	Very large `n_samples`, medium `n_clusters`	`n_clusters` (integer) `batch_size` (integer)
Affinity Propagation [17]	Not scalable with `n_samples`	`damping` (float) `affinity` (categorical)
Mean Shift [8]	Not scalable with `n_samples`	`bin_seeding` (categorical) `bandwidth` (float)
Spectral Clustering [39]	Medium `n_samples`, small `n_clusters`	`n_clusters` (integer) `eigen_solver` (categorical) `affinity` (categorical) `assign_labels` (categorical)
Agglomerative Clustering	Large `n_samples` and `n_clusters`	`n_clusters` (integer) `eigen_solver` (categorical) `affinity` (categorical)
DBSCAN [11]	Very large `n_samples`, medium `n_clusters`	`eps` (float) `min_samples` (integer)
OPTICS [1]	Very large `n_samples`, large `n_clusters`	`min_samples` (integer) `metric` (categorical)

and more ....

10 clustering algorithms in total.

CASH: Loss functions

When labels are not given, how do we quantify the quality of clustering result?

Metric	Range	Normalized Form
Silhouette Coefficient [33]	$-1 \leq x \leq 1$	$(1-x)/2$
Davies-Bouldin Index [9]	$x \geq 0$	$\tanh(x)$
Calinski-Harabasz Index [6]	$x \geq 0$	$1-\tanh(x)$

Basic idea:

If clusters **overlap**, quality is **poor**. If clusters are **well-separated**, quality is **good**.

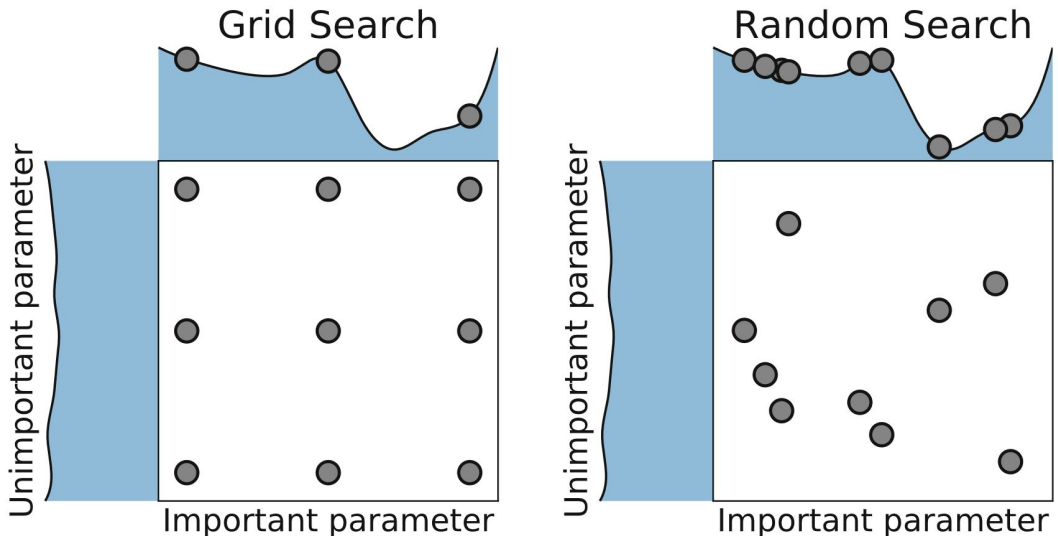
CASH: Loss functions (continued)

Some conditions that we want to discourage. If any of them are met, ***infinity*** *is returned.*

- `number_of_clusters_identified == 1`:
  - To encourage the optimizer to segregate the data points and uncover more patterns.
- $\frac{\#\{\text{points in smallest cluster}\}}{\#\{\text{points in total}\}} < 0.01$:
  - To discourage the optimizer from favoring clustering results with extremely small clusters.
- $\frac{\#\{\text{points in smallest cluster}\}}{\#\{\text{points in largest cluster}\}} < 0.05$:
  - To discourage the optimizer from favoring clustering results with extremely small clusters.

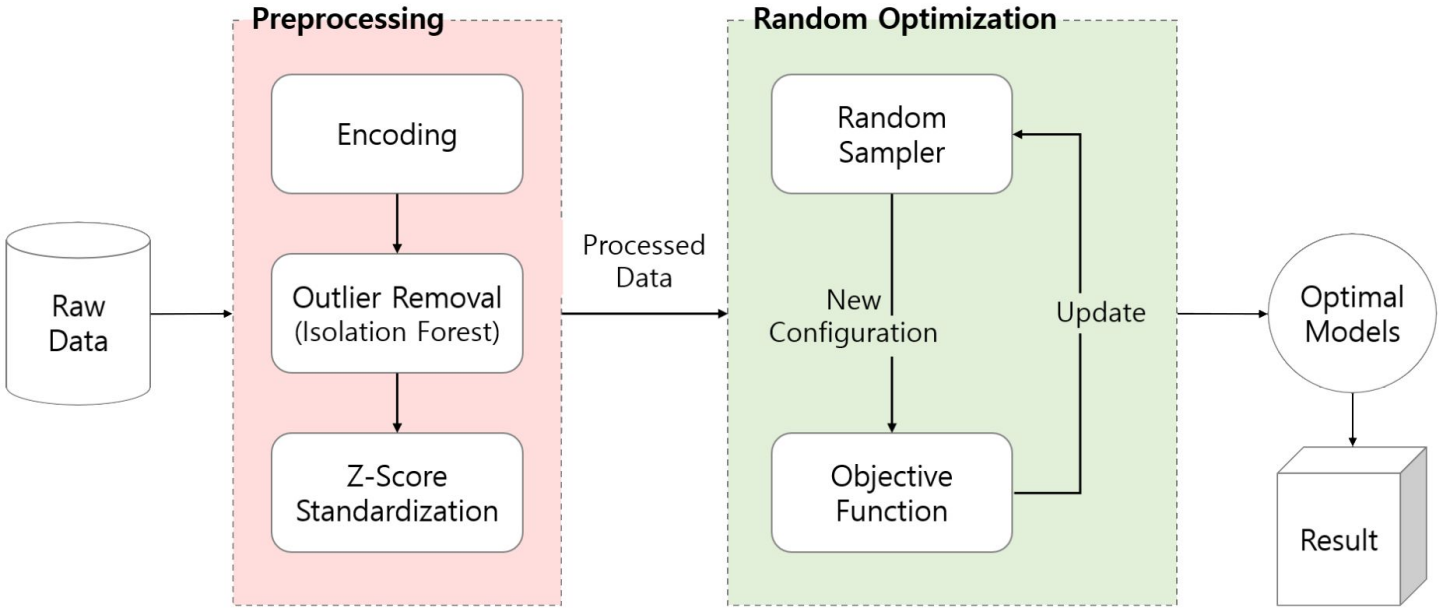
Optimization method 1: Random Search

Grid Search vs Random Search?

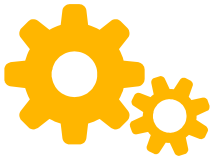


Existing academic research shows that Random Search is superior to Grid Search.

Overview: Random Search



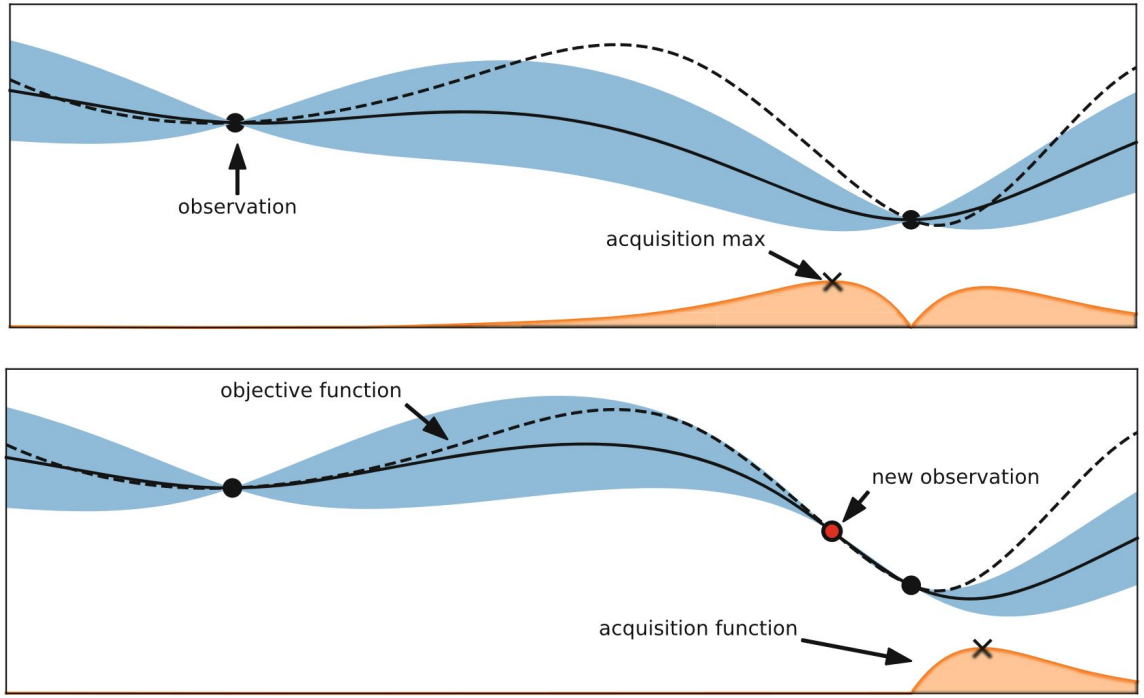
Optimization method 2: Bayesian Optimization



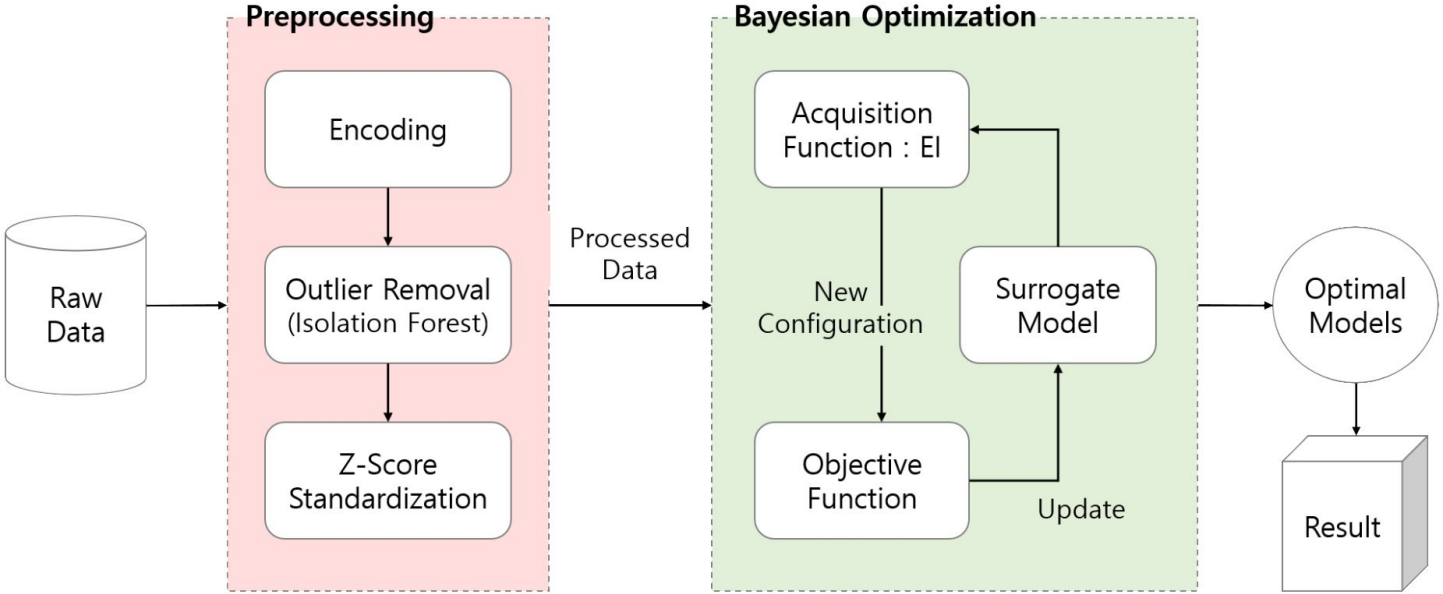
Smarter way to explore the search space.

Objective function	This is the blackbox function, **expensive** to evaluate.
Surrogate model	This model tries to **learn** the objective function from past evaluations. Usually a Gaussian Process (GP) is used.
Acquisition function	This function **recommends** new configurations to be evaluated. Usually Expected Improvement (EI) is used.

Optimization method 2: Bayesian Optimization



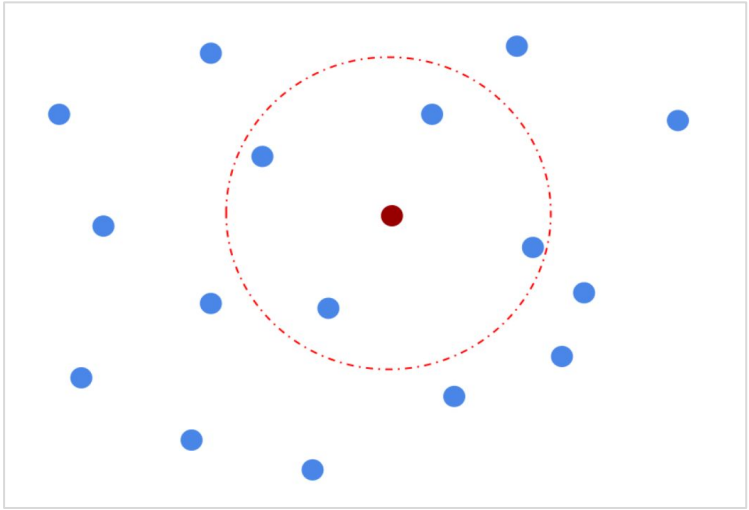
Overview: Bayesian Optimization



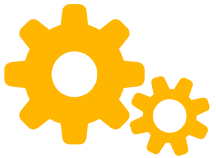
Optimization method 3: Bayesian Optimization + Meta-learning

Meta-learning:

A method to **suggest good configurations** for a novel dataset based on configurations that are known to perform well on similar, previously evaluated, datasets.



Hypothesis: If 2 datasets are similar, their optimal configurations are similar.



Optimization method 3: Bayesian Optimization + Meta-learning

Training a warmstarter via Meta-learning:

1. *Run Bayesian Optimization on a huge variety of benchmark datasets.*
2. *Save the best configurations obtained from those benchmark datasets.*
3. *When given a new dataset, find the **most similar benchmark datasets** to suggest good configurations to warmstart the Bayesian Optimization algorithm.*

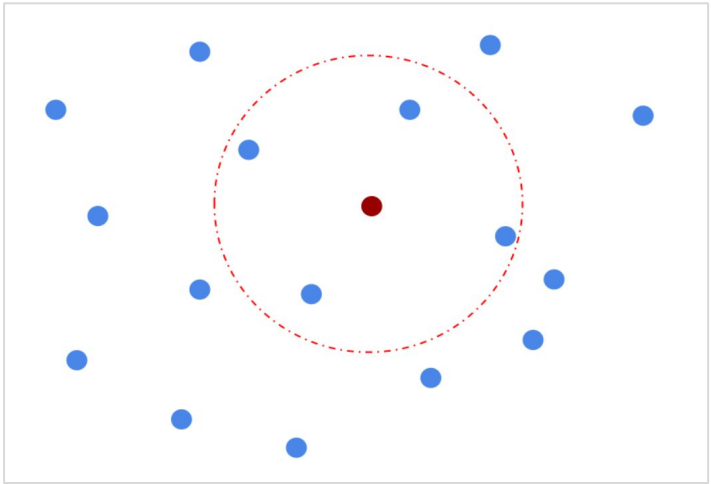
Optimization method 3: Bayesian Optimization + Meta-learning

How to characterize different datasets to identify similar datasets?
*Use **Meta-features!***

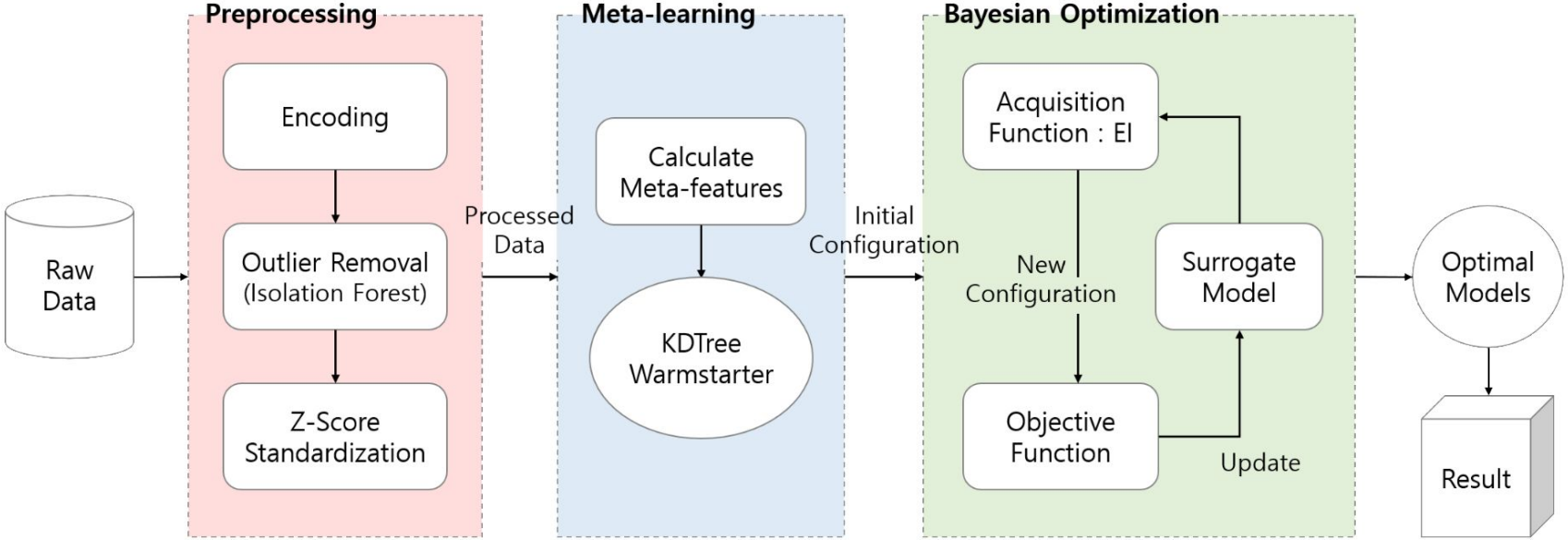
Features	Description	Variants
Number of Instances		log
Number of Features		log, ratio to instances
Number of Missing Values		% missing
Sparsity	$\frac{\#\{\text{missing values \& zeros}\}}{\#\{\text{all values}\}}$	

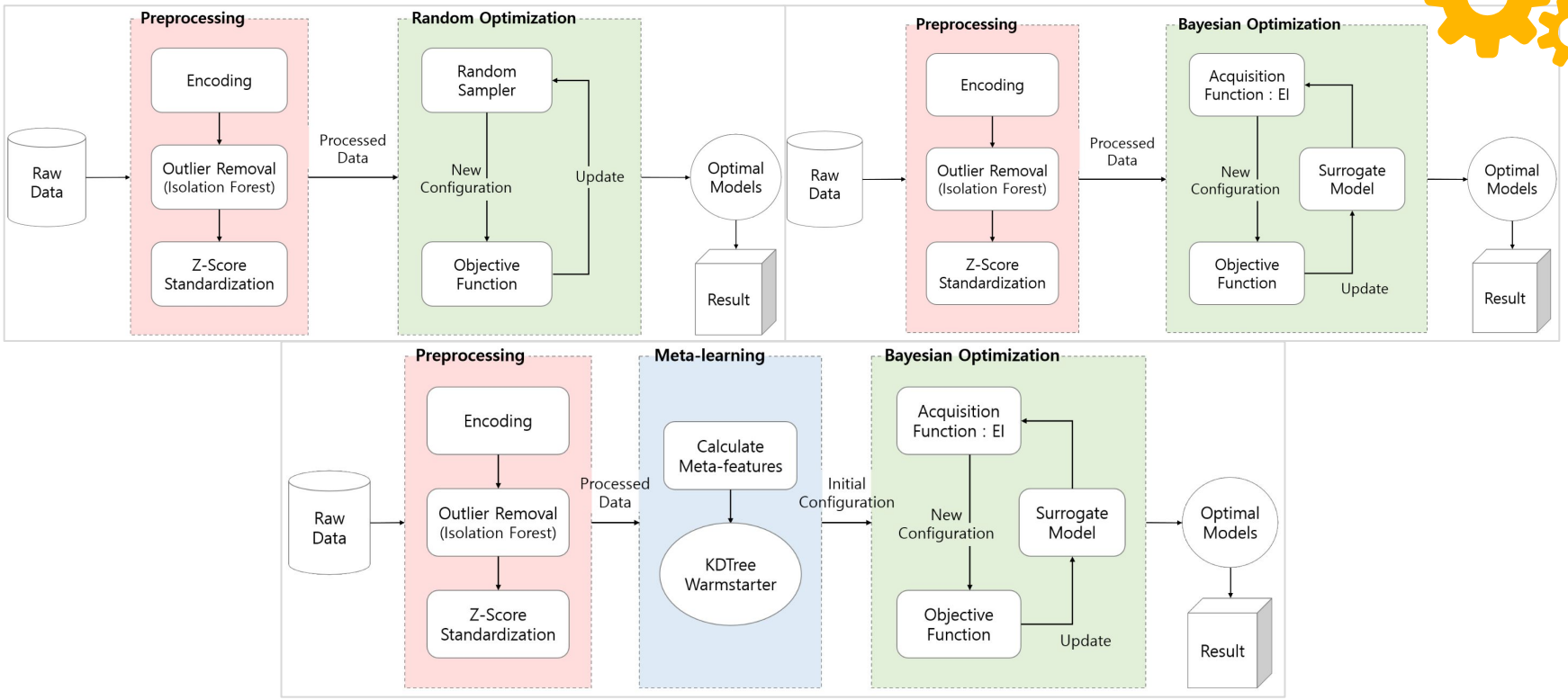
Table 4: General metafeatures for the warmstarter

Features	Description	Variants
Sparsity	$\frac{\#\{\text{zeros}\}}{\#\{\text{numerical values}\}}$	
Skewness	Asymmetry of probability distribution	min, max, median, mean, 1st-quartile, 3rd-quartile
Kurtosis	Tailedness of probability distribution	
Correlation	Dependency between two variables	
Covariance	Joint variability of two variables	



Overview:
Bayesian Optimization + Meta-learning




Preprocessing
Encoding
Outlier Removal (Isolation Forest)
Z-Score Standardization
Raw Data
Processed Data
Random Optimization
Random Sampler
New Configuration
Update
Objective Function
Optimal Models
Result
Preprocessing
Encoding
Outlier Removal (Isolation Forest)
Z-Score Standardization
Raw Data
Processed Data
Bayesian Optimization
Acquisition Function : EI
New Configuration
Surrogate Model
Objective Function
Update
Optimal Models
Result
Preprocessing
Encoding
Outlier Removal (Isolation Forest)
Z-Score Standardization
Raw Data
Processed Data
Meta-learning
Calculate Meta-features
KDTree Warmstarter
Initial Configuration
Bayesian Optimization
Acquisition Function : EI
New Configuration
Surrogate Model
Objective Function
Update
Optimal Models
Result

2

Testing & Results

An experiment to compare the 3 optimization methods.

Experimental Setup: Datasets

120 synthetic datasets.

- S-sets [14]: Synthetic 2-dimensional data with $N = 5000$ vectors and $k = 15$ Gaussian clusters with varying degrees of cluster overlap.
- A-sets [20]: Synthetic 2-dimensional data with increasing number of clusters. There are 150 vectors per cluster.
- Birch-sets [40]: Synthetic 2-dimensional data with $N = 100,000$ vectors and $k = 100$ clusters.
- G2 sets [27]: Gaussian clusters datasets with varying cluster overlap and dimensions.
- DIM-sets (high) [15]: High-dimensional datasets $N = 1024$ and $k = 16$ Gaussian clusters.
- DIM-sets (low) [21]: Low-dimensional synthetic data with Gaussian clusters.
- Unbalance [32]: Synthetic 2-dimensional data with $N = 6500$ vectors and $k = 8$ Gaussian clusters.

Experimental Setup: Train-test split

Bayesian Optimization + Meta-learning approach requires pre-training.

Test: 20 datasets (approx. 16.7%) were randomly selected from the pool of 120 clustering datasets.

Train: The remaining 100 datasets were used to pretrain the warmstarter.

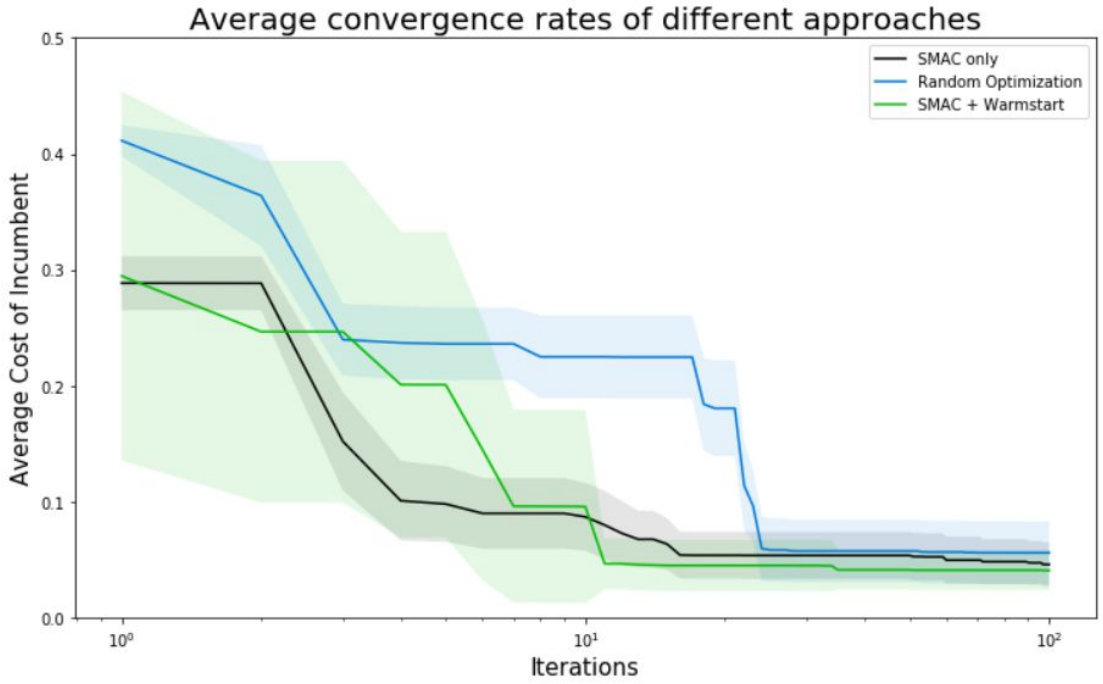
Warmstarter retrieves the top 5 configurations from the top 5 nearest neighbors, giving a total of 25 configurations.

Experimental Setup: Optimization parameters

Optimization parameters (Testing):

- Number of iterations: 100. This is also equivalent to the number of evaluations.
- Cutoff time: 100 seconds. This is the maximum time allowed for evaluating a single configuration.
- Number of folds (k-fold cross validation loss): 3.
- Number of initial configurations from warmstarter: 25. This is only applicable for the SMAC + Warmstarting approach.

Empirical results: average convergence rate



Bayesian Optimization + Meta-learning on S-set

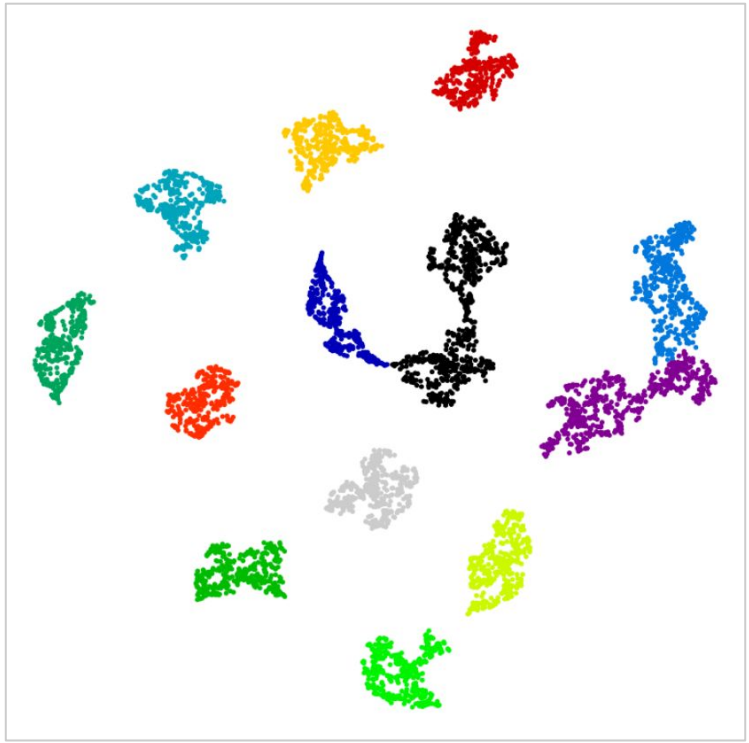


Figure 10: Clustering result on the S-sets (see section 3.1). The dataset comprises of 15 Gaussian clusters in 2-dimensional space with $N = 5000$ points. The optimal configuration obtained by SMAC + Warmstarting consists of a TSNE dimension reduction model + Agglomerative clustering model with `n_clusters = 13`.

Bayesian Optimization + Meta-learning on DIM-set

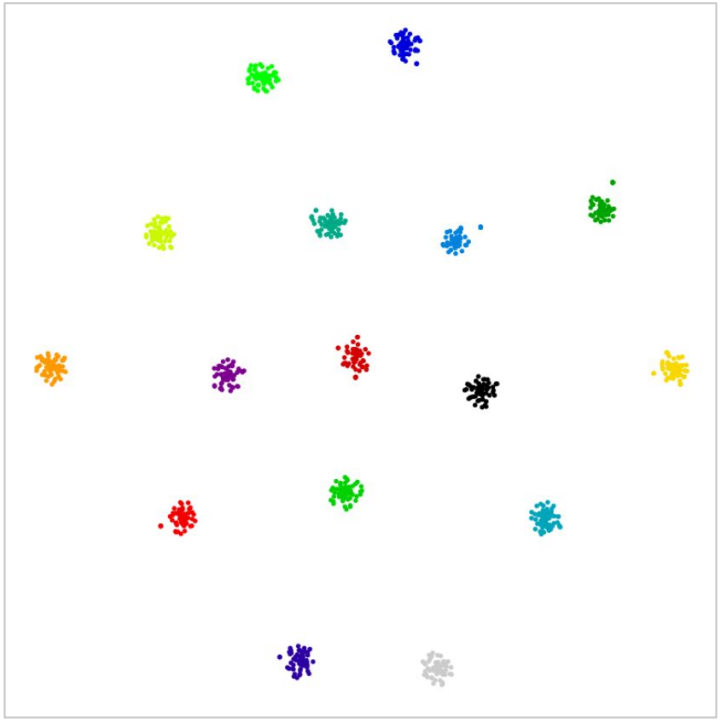


Figure 11: Clustering result on the DIM-sets (see section 3.1). The dataset comprises of 16 Gaussian clusters in 128-dimensional space with $N = 1024$ points. The optimal configuration obtained by SMAC + Warmstarting consists of a Truncated SVD dimension reduction model + Birch clustering model.

Bayesian Optimization + Meta-learning on DIM-set

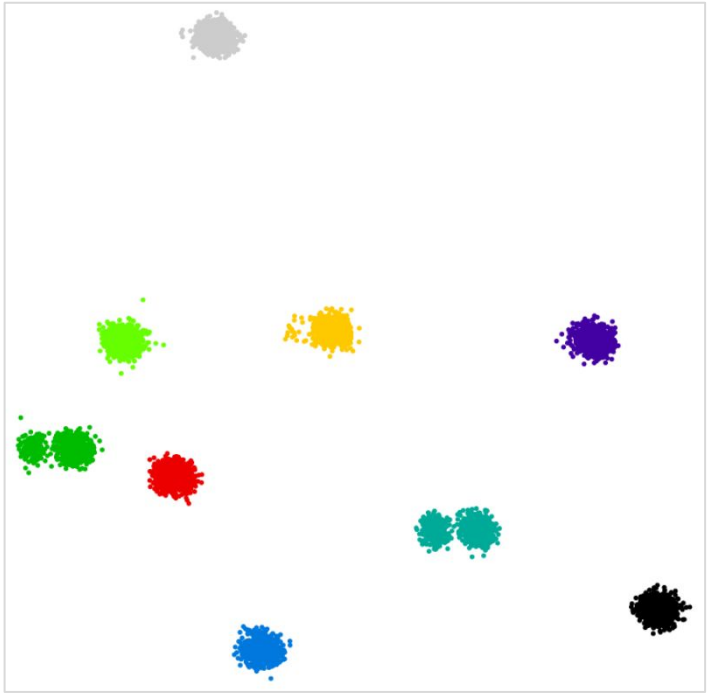
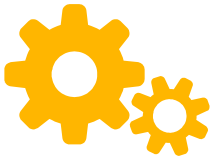


Figure 9: Clustering result on the DIM-sets (see section 3.1). The dataset comprises of 9 Gaussian clusters in 14-dimensional space with $N = 2048$ points. The optimal configuration obtained by SMAC + Warmstarting consists of a PCA dimension reduction model + Affinity Propagation clustering model.

3

Conclusion: Limitations and Further Research

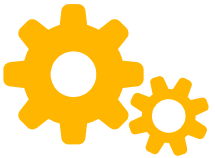


Dimension Reduction

Dimension reduction algorithm and hyperparameters were included into the bayesian optimization process

Reasoning:

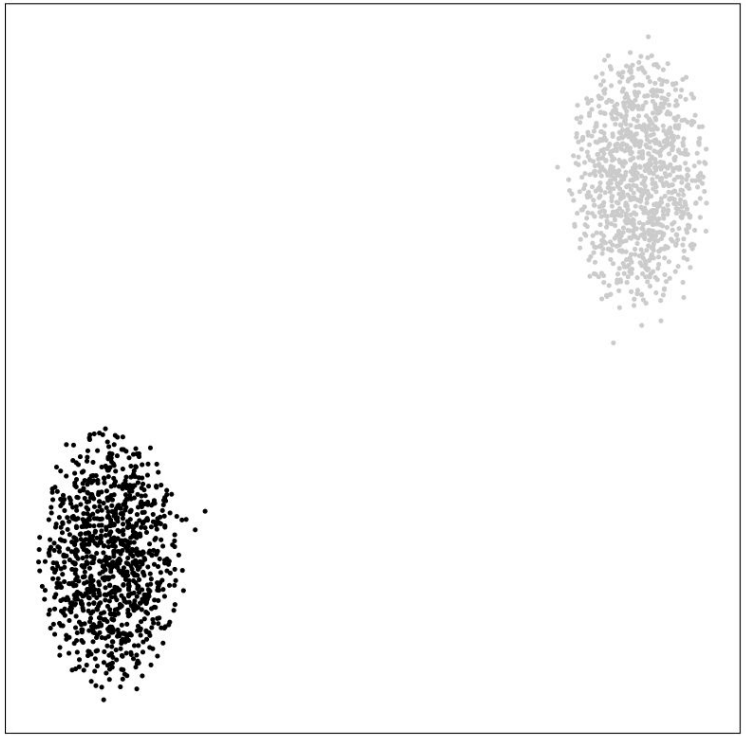
- Dimension reduction generally accepted to make the entire pipeline more efficient.
- Allows for clearer visualization by the end user.
- Allows clustering evaluation metrics to perform better.

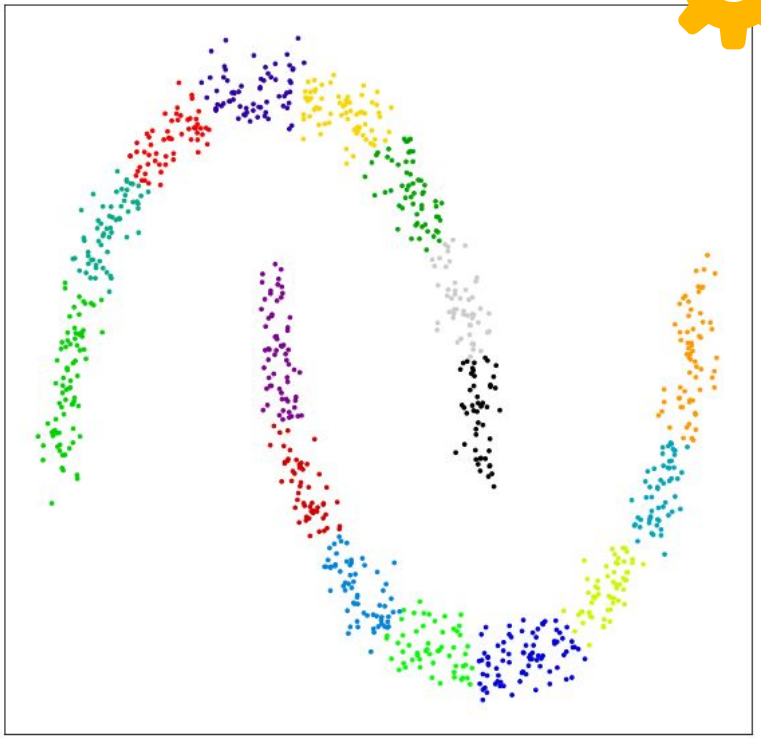


Clustering Metrics

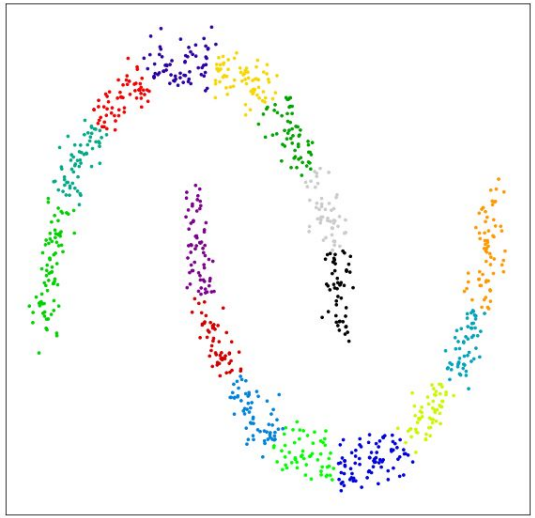
The clustering metrics all favor convex clustering.

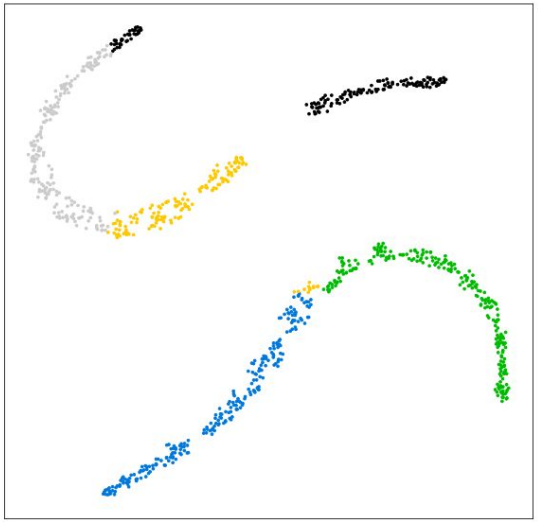
But this means that the clustering algorithm does not perform well on non-convex

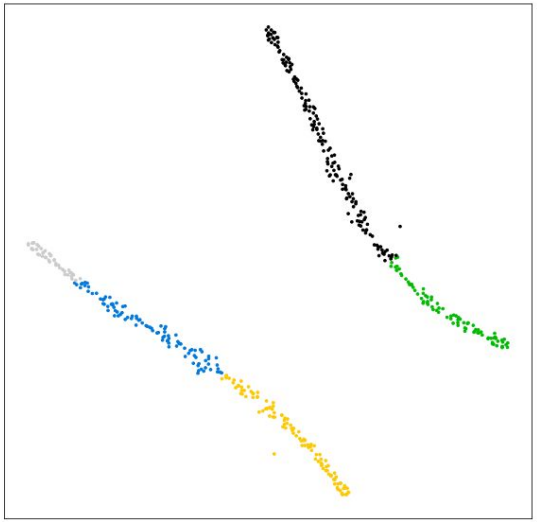


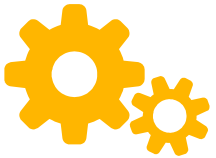


Limitations





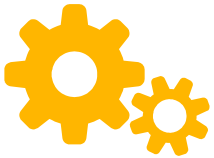




Topological data analysis

Topological data analysis may solve the issue of evaluating non-convex clustering.

However, because these tools do not scale well with dataset size, we opted

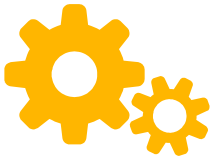


Scalability issues

For larger datasets, evaluation can take very long.

Possible Fixes:

- Subsampling
- Multi-fidelity evaluation



Unanswered Questions

Is real world data convex?

`

4

Q and A