Evaluation Toolkit for Filtered ANN:

Metrics and Experiments on Variable Filter Specificities and Attribute Distributions

Abderrahmane Baidoune
College of Computing, UM6P
School of Computer Science **Supervisors:**
Professor Karima Echihabi
TA Anass Ait Omar

December 29, 2024

Contents

1 Methodology **3**
- 1.1 Data Generation . . . 3
  - 1.1.1 Vector Generation . . . 3
  - 1.1.2 Attribute Generation . . . 3
  - 1.1.3 Implementation Details . . . 4
- 1.2 Dataset Configuration and Generation . . . 4
  - 1.2.1 Dataset Configurations . . . 4
  - 1.2.2 Dataset Generation Process . . . 5
- 1.3 Query Workload Construction . . . 5
  - 1.3.1 Filter Specificity Levels . . . 5
  - 1.3.2 Query Vector Generation . . . 5
- 1.4 Index Construction . . . 6
  - 1.4.1 Index Building Process . . . 6
- 1.5 Filtering Strategies . . . 6
  - 1.5.1 Post-Filtering . . . 6
  - 1.5.2 Inline Filtering . . . 6
- 1.6 Evaluation Metrics . . . 7
  - 1.6.1 Query Time . . . 7
  - 1.6.2 Recall . . . 7
  - 1.6.3 Filter Friction . . . 7

2 Results and Analysis **8**
- 2.1 Overall Performance . . . 8
  - 2.1.1 Query Time . . . 8
  - 2.1.2 Recall . . . 8
  - 2.1.3 Filter Friction . . . 9
- 2.2 Performance by Attribute Distribution . . . 9
  - 2.2.1 Analysis . . . 9
- 2.3 Performance by Filter Specificity . . . 9
  - 2.3.1 Analysis . . . 10
- 2.4 Performance by Dataset Size . . . 10
  - 2.4.1 Analysis . . . 10
  - 2.4.2 Performance Degradation . . . 11
- 2.5 Performance by Filter Method . . . 11
  - 2.5.1 Analysis . . . 11
  - 2.5.2 Recall by Specificity and Method . . . 11
- 2.6 Visual Analysis . . . 11
  - 2.6.1 Filter Friction Analysis . . . 12

2.6.2 Filtering Methods Comparison 12
2.6.3 Correlation Matrix 12
2.6.4 Parameter Relationships 13

3 Recommendations and Conclusion **14**

Chapter 1

Methodology

1.1 Data Generation

The data generation involved creating high-dimensional vectors and corresponding binary attributes under distinct distribution patterns to emulate diverse real-world scenarios.

1.1.1 Vector Generation

Each data point in the dataset comprises a 64-dimensional vector. The vectors are generated using two primary distribution methods:

- **Normal Distribution**: Vectors are sampled from a standard normal distribution with a mean of 0 and a standard deviation of 1. This distribution ensures that the vectors are centered around the origin with symmetric variance in all dimensions.
- **Uniform Distribution**: Alternatively, vectors can be generated from a uniform distribution ranging between -1 and 1. This approach provides a balanced spread of vector components across all dimensions.

The choice of distribution affects the clustering and proximity of vectors in the high-dimensional space, thereby influencing the performance of the ANN search.

1.1.2 Attribute Generation

Accompanying each vector is a set of 10 binary attributes. These attributes are generated under three distinct distribution paradigms to simulate varying levels of complexity and correlation:

1. **Uniform Distribution**: Each attribute is equally likely to be present (1) or absent (0). This scenario represents a balanced attribute presence, facilitating straightforward filtering without inherent biases.
2. **Skewed Distribution**: Certain attributes are significantly more prevalent than others. This skew introduces imbalance, where some attributes dominate, potentially affecting the selectivity and efficiency of filtering mechanisms.
3. **Correlated Attributes**: Specific attributes co-occur frequently, introducing interdependencies among attributes. This correlation complexity can exacerbate filter friction, as satisfying multiple correlated attributes may require traversing more extensive regions of the vector space.

1.1.3 Implementation Details

The data generation process leverages probabilistic models to achieve the desired distributions:

- For **uniform** and **skewed** distributions, binary attributes are generated using binomial sampling with specified probabilities. In the skewed scenario, a Beta distribution governs the probability of an attribute being active, introducing variability and imbalance.
- For **correlated attributes**, a covariance matrix is constructed to induce correlations among attributes. This matrix is derived from a random symmetric matrix with controlled eigenvalues to ensure positive definiteness. Multivariate normal sampling followed by thresholding at 0.5 transforms the continuous data into binary attributes while preserving the specified correlations.

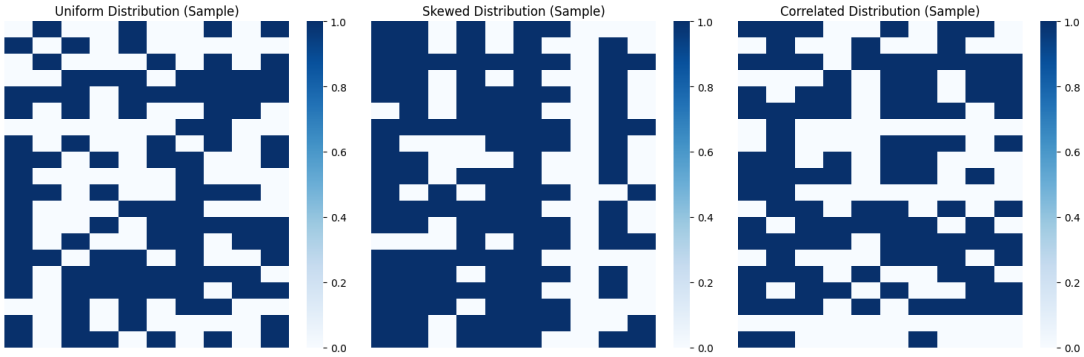

Figure 1.1: Sample Attribute Distributions

Description: Figure 1.1 showcases heatmaps representing the attribute distributions for uniform, skewed, and correlated scenarios.

1.2 Dataset Configuration and Generation

1.2.1 Dataset Configurations

Three distinct dataset sizes are configured to assess scalability:

- **Small**: 1,000 samples
- **Medium**: 5,000 samples
- **Large**: 10,000 samples

Each dataset comprises 64-dimensional vectors and 10 binary attributes, ensuring uniformity in vector dimensionality and attribute count across different dataset sizes.

1.2.2 Dataset Generation Process

For each dataset size and attribute distribution type, the following steps are undertaken:

1. **Vector Generation**: Depending on the specified distribution type (normal or uniform), 64-dimensional vectors are generated to form the dataset's feature space.
2. **Attribute Generation**: Based on the distribution type (uniform, skewed, or correlated), 10 binary attributes are generated for each vector. This involves:
    - Utilizing binomial sampling for uniform and skewed distributions, with the skewed distribution governed by a Beta distribution to introduce attribute prevalence variability.
    - Constructing a covariance matrix for correlated attributes, ensuring positive definiteness through eigenvalue adjustments. Multivariate normal sampling followed by thresholding at 0.5 is employed to generate correlated binary attributes.
3. **Dataset Assembly**: The generated vectors and attributes are collated into structured datasets categorized by size and attribute distribution type.

1.3 Query Workload Construction

To evaluate the filtering mechanisms,I constructed a diverse set of queries, each accompanied by specific attribute constraints.

1.3.1 Filter Specificity Levels

Queries are categorized based on the number of attribute constraints they impose:

1. **Low Specificity**: Queries with a minimal number of attribute constraints (e.g., 1 to 2 attributes). These queries are less restrictive, allowing a broader set of vectors to satisfy the filter criteria.
2. **Mixed Specificity**: Queries with a moderate number of attribute constraints (e.g., 3 to 5 attributes). This level introduces a balance between selectivity and inclusiveness.
3. **High Specificity**: Queries with a high number of attribute constraints (e.g., 5 to 10 attributes). These queries are highly restrictive, significantly narrowing down the pool of vectors that meet the filter conditions.

1.3.2 Query Vector Generation

For each specificity level, query vectors are generated independently:

- Vectors are sampled from the same distribution as the dataset vectors (normal or uniform) to maintain consistency.
- Corresponding attribute filters are randomly selected based on the desired specificity, ensuring a representative and unbiased distribution of query constraints.

1.4 Index Construction

The HNSW index was chosen for Approximate Nearest Neighbor (ANN) searches.

1.4.1 Index Building Process

For each dataset variant (size and attribute distribution), the following steps are undertaken to construct the HNSW index:

1. **Initialization**: The HNSW index is initialized with the specified $ef\_construction$ and M parameters, establishing the foundational structure of the graph.
2. **Data Ingestion**: All vectors from the dataset are added to the index, each assigned a unique label to facilitate retrieval and filtering.
3. **Parameter Configuration**: Post-ingestion, the ef parameter is set to the $initial\_ef$ value, preparing the index for subsequent search operations.

1.5 Filtering Strategies

I used two filtering methods to compare between them

1.5.1 Post-Filtering

Post-Filtering involves executing the ANN search first and then applying attribute-based constraints to the retrieved results. The process is as follows:

1. **ANN Search**: Perform a k-NN search to retrieve a predefined number of nearest neighbors (e.g., $2k$) without considering any attribute constraints.
2. **Attribute Filtering**: Iterate through the retrieved neighbors and retain only those that satisfy the specified attribute filter conditions.

1.5.2 Inline Filtering

Inline Filtering integrates attribute-based constraints directly into the search process, dynamically adjusting search parameters to accumulate valid results. The methodology emulates native inline filtering capabilities, which is not directly supported by the HNSW library used.

1. **Initial Search Parameter**: Begin with an initial ef value ($initial\_ef$) to control the breadth of the search.
2. **Candidate Retrieval**: Execute the ANN search with the current ef value to retrieve a set of candidate neighbors.
3. **Attribute Filtering**: Apply the attribute filter to the retrieved candidates, accumulating valid results.

4. **Iterative Expansion**: If the desired number of filtered results (k) is not met, iteratively increase the ef parameter (e.g., doubling it) up to a predefined max_ef limit to retrieve more candidates.

5. **Termination**: The process continues until k valid results are obtained or the max_ef threshold is reached.

1.6 Evaluation Metrics

1.6.1 Query Time

Query Time measures the duration taken to execute a query and retrieve the desired results. It encompasses the entire process from initiating the ANN search to obtaining the final filtered results.

- **Measurement**: Recorded in seconds, it captures the latency introduced by the filtering method.
-

1.6.2 Recall

Recall quantifies the accuracy of the ANN search by determining the proportion of relevant items retrieved compared to the total relevant items present in the ground truth.

- **Calculation**:
$$\text{Recall} = \frac{|\text{Retrieved Relevant Items}|}{|\text{Total Relevant Items}|}$$
-

1.6.3 Filter Friction

Filter Friction is a composite metric reflecting the selectivity of the attribute filter and the correlation among filtered attributes. It encapsulates the complexity introduced by filtering constraints.

- **Components**:
  - **Selectivity**: The proportion of the dataset that satisfies the filter conditions.
  - **Correlation**: The degree of interdependence among the specified attributes.
- **Calculation**:
$$\text{Filter Friction} = (1 - \text{Selectivity}) \times (1 + \text{Average Correlation})$$
  where:
  - Selectivity is the proportion of the dataset satisfying the filter conditions.
  - Average Correlation is the mean of the absolute Pearson correlation coefficients among all pairs of specified attributes.
- **Significance**: Higher filter friction scores indicate more challenging filtering conditions, potentially leading to increased query times and reduced recall.

Chapter 2

Results and Analysis

2.1 Overall Performance

The evaluation encompasses three primary metrics: Query Time, Recall, and Filter Friction. These metrics provide a multifaceted view of the HNSW index's efficiency and accuracy.

2.1.1 Query Time

- **Mean**: 0.0026 seconds
- **Standard Deviation**: 0.0059 seconds
- **Minimum**: 0.0001 seconds
- **Maximum**: 0.0622 seconds

Interpretation: The average query time of approximately 2.6 milliseconds indicates that the HNSW index operates efficiently for most queries. However, the significant standard deviation suggests variability in query performance.

2.1.2 Recall

- **Mean**: 0.3433
- **Standard Deviation**: 0.3946
- **Minimum**: 0.0000
- **Maximum**: 1.0000

Interpretation: A mean recall of 34.33% reflects that, on average, approximately one-third of the relevant items are retrieved.

2.1.3 Filter Friction

- **Mean**: 0.8334
- **Standard Deviation**: 0.2947
- **Minimum**: 0.0345
- **Maximum**: 1.2299

Interpretation: The average filter friction score of approximately 0.8334 suggests a moderate level of complexity in satisfying the filter conditions. Higher friction scores indicate more challenging filtering scenarios.

2.2 Performance by Attribute Distribution

Attribute distribution plays a crucial role in the efficiency and accuracy of ANN searches. The evaluation considered three distinct distributions: Uniform, Skewed, and Correlated.

Table 2.1: Performance Metrics by Attribute Distribution

Distribution Type	Query Time (s)	Recall	Filter Friction
Correlated	0.003867	0.340451	0.919132
Skewed	0.002204	0.325347	0.780923
Uniform	0.001856	0.364211	0.800240

2.2.1 Analysis

Uniform Distribution exhibited the **fastest query times** and the **highest recall** among the three distributions. This performance can be attributed to the balanced nature of uniform attributes, which facilitates more efficient filtering and retrieval processes. **Skewed Distribution** resulted in moderately increased query times and slightly lower recall compared to the uniform distribution. The imbalance introduced by skewed attributes likely necessitates more extensive search operations to satisfy filter conditions, thereby impacting performance.

Correlated Distribution showed the **highest filter friction** and the **slowest query times**. The interdependencies among attributes complicate the filtering process, requiring the algorithm to traverse more extensive regions of the vector space to locate relevant items.

2.3 Performance by Filter Specificity

Filter specificity determines the restrictiveness of the attribute constraints applied during the search. The evaluation categorized filter specificity into Low, Mixed, and High.

Table 2.2: Performance Metrics by Filter Specificity

Filter Specificity	Query Time (s)	Recall	Filter Friction
High	0.004865	0.029340	1.058194
Low	0.000549	0.801042	0.488284
Mixed	0.002513	0.199628	0.953817

2.3.1 Analysis

High Specificity: Queries with high specificity imposed stringent attribute constraints, resulting in the lowest recall of approximately 2.93% and the highest filter friction score. The substantial increase in query time indicates that the search algorithm struggles to locate sufficient relevant items under such restrictive conditions.

Low Specificity: These queries imposed minimal attribute constraints, achieving a high recall of approximately 80.10% and the lowest filter friction score. The query times were minimal, reflecting the ease of satisfying less restrictive filters.

Mixed Specificity: Representing intermediate levels of filter restrictiveness, mixed specificity queries achieved a moderate recall of approximately 19.96% and a substantial filter friction score. Query times were also intermediate, balancing the challenges of moderate filter constraints.

2.4 Performance by Dataset Size

Scalability is a critical factor in evaluating the HNSW index's practicality for large-scale applications. The evaluation considered three dataset sizes: Small, Medium, and Large.

Table 2.3: Performance Metrics by Dataset Size

Dataset Size	Query Time (s)	Recall	Filter Friction
Large	0.004228	0.324306	0.826782
Medium	0.002644	0.338542	0.839655
Small	0.001055	0.367163	0.833857

2.4.1 Analysis

Small Dataset: Achieved the fastest query times and the highest recall, indicating efficient search operations in smaller datasets where the algorithm can swiftly navigate the vector space to locate relevant items.

Medium Dataset: Query times and recall slightly decreased compared to the small dataset, reflecting the increased complexity and search space inherent in larger datasets.

Large Dataset: Demonstrated the highest query times and the lowest recall among the three sizes, underscoring the challenges of maintaining performance and accuracy as the dataset scales.

2.4.2 Performance Degradation

The performance degradation from small to large datasets was quantified at approximately 4.01x increase in query time. This linear scaling suggests that while the HNSW index maintains reasonable efficiency as data grows, there is a notable impact on query performance and recall.

2.5 Performance by Filter Method

The evaluation compared two distinct filtering methods: Inline Filtering and Post-Filtering.

Table 2.4: Performance Metrics by Filter Method

Filter Method	Query Time (s)	Recall	Filter Friction
Inline	0.004786	0.338641	0.834598
Post	0.000498	0.348032	0.832265

2.5.1 Analysis

Inline Filtering: Exhibited higher query times and comparable recall to post-filtering. The increased query time is attributable to the iterative adjustment of the ef parameter to satisfy filter constraints, which inherently demands more computational resources.

Post-Filtering: Demonstrated significantly lower query times with a marginally higher recall. The streamlined process of performing filtering after the ANN search reduces computational overhead, enhancing efficiency without substantially compromising accuracy.

2.5.2 Recall by Specificity and Method

```
filter_specificity  filter_method
high                inline          0.026389
                    post            0.032292
low                 inline          0.792361
                    post            0.809722
mixed               inline          0.197173
                    post            0.202083
```

Interpretation: Across all filter specificities, post-filtering consistently outperforms inline filtering in terms of recall. This suggests that the post-filtering approach is more effective in retrieving relevant items, especially under less restrictive and moderately restrictive filter conditions.

2.6 Visual Analysis

The accompanying visualizations provide a deeper understanding of the quantitative results:

2.6.1 Filter Friction Analysis

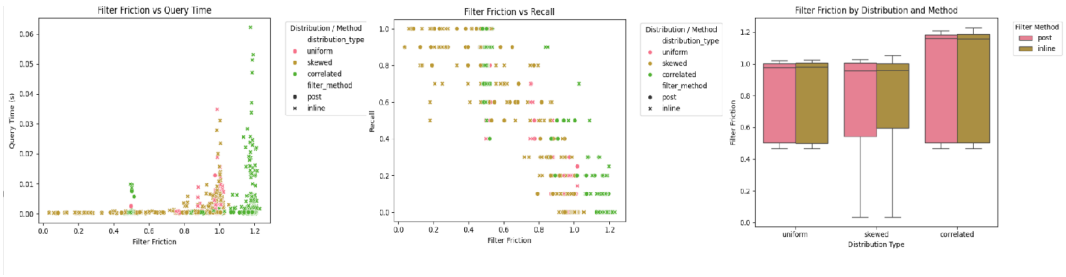

Figure 2.1: Filter Friction Analysis

2.6.2 Filtering Methods Comparison

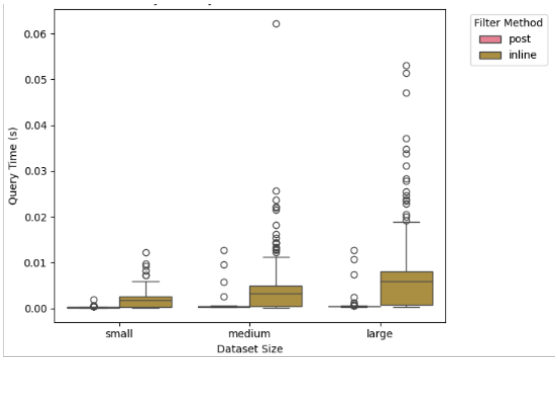

Figure 2.2: Query time by filtering method and dataset size

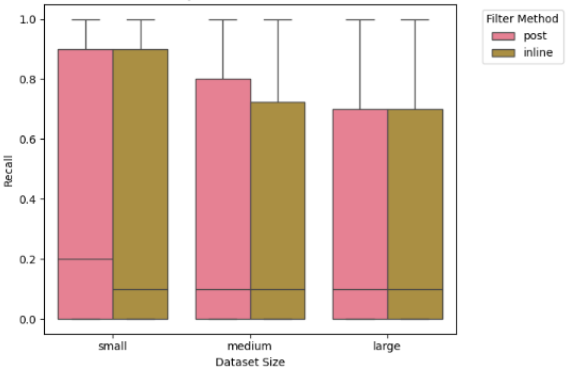

Figure 2.3: Recall by filtering method and dataset size

2.6.3 Correlation Matrix

the relationships between key HNSW parameters (ef construction, M, initial ef, max ef) and their impact on performance metrics.

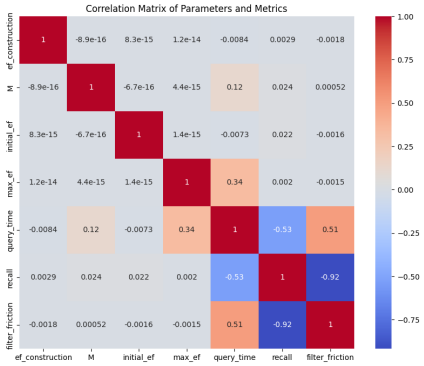

Figure 2.4: Correlation Matrix of Parameters and Metrics

Description: Figure 2.4 displays a heatmap representing the correlations between HNSW parameters and performance metrics. Notably, there is a strong negative correlation between recall and filter friction (-0.92), indicating that higher filter friction significantly hampers recall. A moderate positive correlation (0.51) exists between query time and filter friction, signifying that more complex filters extend query durations. The correlations between construction parameters (ef_construction, M) and metrics are relatively weak, suggesting that these parameters have a limited direct impact on recall and filter friction.

2.6.4 Parameter Relationships

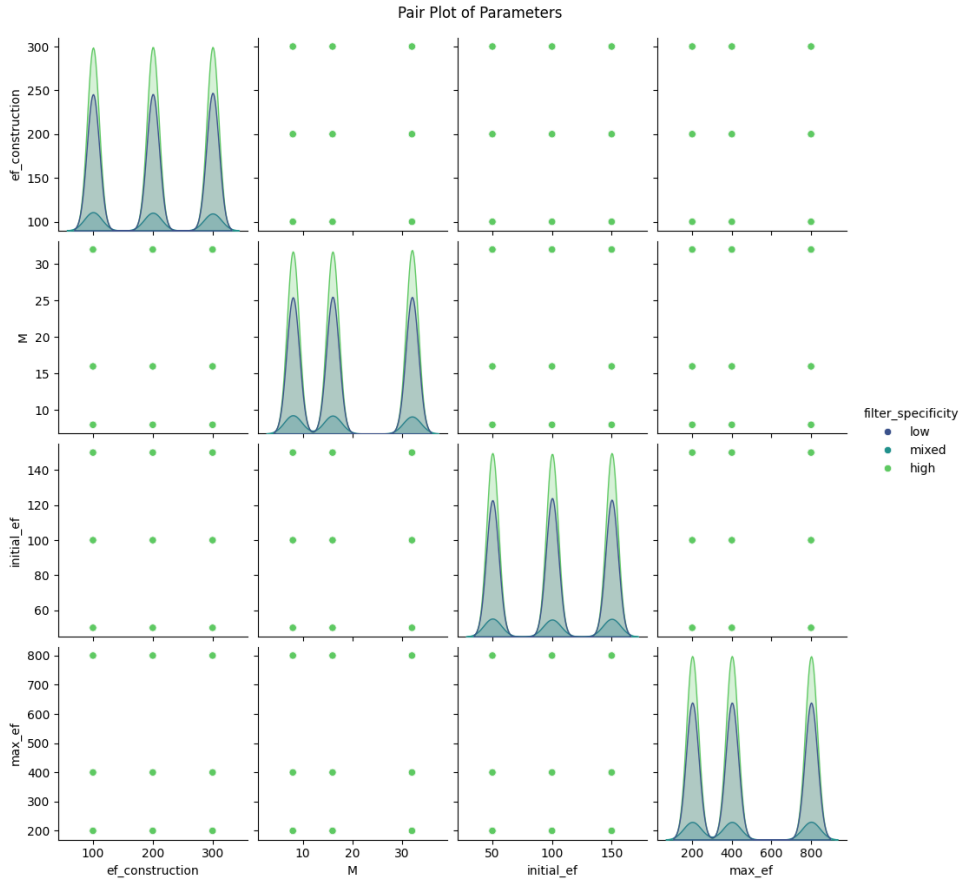

Figure 2.5: Pair Plot of HNSW Parameters by Filter Specificity

Chapter 3

Recommendations and Conclusion

Recommendations:

- **Parallel Computation**: Incorporate multi-threading or distributed computing solutions to handle larger datasets more efficiently, thereby mitigating the scaling challenges observed in query time and recall.
- **Adaptive Index Tuning**: Dynamically adjust HNSW parameters (`ef_construction`, `M`) based on evolving data distributions and query behaviors to maintain an optimal balance between speed and accuracy.
- **Feature Engineering**: For correlated attributes, explore dimensionality reduction or decorrelation methods to reduce filter friction and improve recall.
- **Hybrid Filtering Approach**: Combine post-filtering for efficiency and inline filtering for targeted queries requiring higher precision, leveraging the strengths of each method.

Conclusion: This detailed evaluation emphasizes the key factors influencing the HNSW index's performance in filtered ANN scenarios, notably attribute distributions, filter specificity, scalability, and filtering methods. Post-filtering generally offers superior efficiency and recall, while inline filtering may be reserved for specialized use cases. However, as dataset size and complexity grow, employing parameter optimization, parallel computation, and thoughtful feature engineering becomes imperative to sustain robust performance.
And for more plots and anlysis of the parameters tunning, check the markdowns in the notebook:).

Personal Experience and Use of GPT: Initially, I was focusing on text-based datasets, but I later shifted to generating synthetic numerical data with varying attribute distributions (uniform, skewed, and correlated). Throughout the project, I used GPT to:

- Clean and refactor code, ensuring consistent documentation and streamlined function structures.
- Generate and refine LaTeX report sections, maintaining a professional and cohesive format.
- Suggest solutions for data generation, parameter tuning, and methodological improvements when shifting from text-based to numerically generated data.