Smart Query Sampling with Feature Coverage and Unsupervised Machine Learning

Jerry Tang
Stanford University
Stanford, USA
jhtang21@stanford.edu

Ryan Druckman
Snowflake Computing
Bellevue, USA
ryan.druckman@snowflake.com

Louis Magarshack
Snowflake Computing
San Mateo, USA
louis.magarshack@snowflake.com

Nate McNamara
Snowflake Computing
New York, USA
nate.mcnamara@snowflake.com

Jim Ahn
Snowflake Computing
San Mateo, USA
jim.ahn@snowflake.com

Jiaqi Yan
Snowflake Computing
San Mateo, USA
jiaqi.yan@snowflake.com

***Abstract*—Snowflake is a Data Cloud Platform with billions of daily customer queries. As Snowflake continues to grow with larger query volume, more diverse customers, and frequent software upgrades, there is a higher than ever demand for efficient testing. Snowtrail is a Snowflake-internal testing infrastructure that prepares, executes, and measures workloads from eligible customer queries. One of Snowtrail's challenges is to generate concise workloads that are both representative of customer queries and thorough in testing Snowflake software features. Existing Snowtrail query workloads are either explicitly provided by users or pseudo-randomly selected. Here, we provide a novel sampling solution based on clustering feature-vectorized queries with unsupervised tasks. Variance across columns and multivariate Kullback–Leibler divergence scores show that a small K-Means sampled workload contains more spread but is less similar to the original population distribution. Column-wise Kolmogorov–Smirnov tests shows a general distinction between the K-Means sampled population from the true distribution regardless of sample size. Runtime experiments confirm that K-Means-based sampling is likely at least quadratic time with regard to the population size and linear time with regard to the sample size, making it not as practical for overly large datasets. Finally, an elbow method analogous to the K-Means elbow method is implemented to select the ideal sample size based on balancing variance and similarity to the population. Overall, this work brings practical improvements to the Snowflake Data Cloud, and the approach described in the paper is generally applicable to data management systems with feature collection capabilities.**

***Index Terms*—cloud platform, Snowflake, machine learning, clustering, sampling**

I. Introduction

The Cloud [1] has moved data storage and software execution away from local servers and towards data centers, such as services offered by Amazon, Google, and Microsoft. Due to shared infrastructure, cloud computing allows the redistribution of resources and services by flexible demand. From a customer standpoint, cloud database has the advantage of higher storage capacity and accessing capabilities, as well as lower costs and the ease to manage and analyze [2]. Building on the foundation of cloud providers is the growth of cloud native Software-as-a-Service (SaaS) and Platform-as-a-Service (PaaS) models, such as Salesforce, Twilio, and Google Workspace.

Another benefit cloud computing enables is the possibility to perform online software updates without service downtime. As cloud services operate globally and around the clock, this capability has become a critical feature. The Snowflake Data Cloud [3] is a pay-as-you-go data PaaS that is built bespoke for the cloud. Snowflake fundamentally dissociates data computations from storage, making queries more elastic and highly available. The more frequently a cloud-based service provider updates its systems, however, the less time available for rigorous testing. To unleash the innovation potential of cloud-based systems, it requires the identification, construction, and execution of testing workloads that optimize the balance of thoroughness and concision. Snowtrail [4] is a component in Snowflake's testing pipeline that generates testing query workloads from customer queries. This infrastructure has the objective to generate testing workloads that extensively tests Snowflake features, consisting of queries that are representative of popular customer use cases. By leveraging customer queries to do regression testing, Snowtrail is able to ensure that customer workloads are less likely to be negatively impacted by new releases.

Meanwhile, the evolution of machine learning algorithms, along with the improved availability datasets, allows for sophisticated techniques in sample selection tasks. In cross discipline studies, both supervised and unsupervised machine learning has found applications in areas such as genes selection [5], crop selection [6], and features selection [7]. Specifically, clustering algorithms have been applied to produce samples with items that are representative of different classifications. Cluster-based sampling is especially useful in pre-processing imbalanced datasets, by clustering and selecting fixed amounts of representative samples from each class [8] [9]. Following this intuition, this work aims to use cluster-based sampling to generate concise workloads with varied queries based on Snowflake features.

979-8-3503-1635-3/23/$31.00 ©2023 IEEE
DOI 10.1109/FiCloud58648.2023.00036

II. Relevant Work

A. Current Snowtrail Sampling Procedures

Snowtrail workloads can be generated in one of two ways: developers can either provide an explicit set of queries to replay, or allow Snowtrail to sample queries. In the first approach, developers are responsible for determining the set of queries they wish to run with Snowtrail. Using our internal data analytical system built on top of the Snowflake Data Cloud itself, which collects metadata about all customer queries, developers can filter and join rich data sets to collect queries and build workloads that target specific use-cases, features, or query types. The alternative approach is to let Snowtrail sample queries randomly. Snowtrail itself has some constraints around which queries it can replay (which statement types Snowtrail supports, etc), and developers can provide additional constraints (queries with specific features, queries from specific dates, etc). Snowtrail then randomly samples a specified number of queries from those satisfying the provided constraints.

B. NLP-based Workload Generalization

Previous research at Snowflake has investigated queries based on raw SQL statements and optimized Snowflake query plans [10]. It was shown that representation learning, e.g. Query2Vec, can be used to vectorize queries. Then applying methods such as K-Means would allow clustering and workload summarizing. Furthermore, error-generating queries can be classified to find common patterns, potentially providing a solution for testing and errors prediction.

Although novel, this NLP based technique has two main drawbacks with respect to Snowtrail applications. Firstly, SQL text and query plans are too high level when it comes to making predictions and clustering based on Snowflake-secific features. For example, semantically similar queries can have significantly different features and applicable optimizations, which weakens the correlation between query expressions and specific errors. Secondly, computation for representation learning over large datasets is expensive and time consuming, and Snowflake's frequent software updates require frequent re-learning based on queries of the newest Snowflake version.

III. K-Means Sampling Procedure

Our sampling procedure contains three main steps: retrieving feature-tracking queries within Snowflake, vectorizing and balancing the dataset, and running a series of unsupervised tasks to sample queries. In addition to using predetermined parameters, the Snowtrail user also has the option to let our algorithm determine the optimal sample size. Fig. 1 is a visualized flowchart for this procedure.

A. Features Tracking

It is important to be able to correlate job performance with the usage of different Snowflake features. For this reason, we built a custom table that joins together each customer query with performance metrics such as: compile time, execution time, queue time among others with features that could impact performance. Features that might impact performance fall into 3 main categories: characteristics of the data, characteristics of the query, and applied optimizations. Data characteristics include: the number of tables, how they're joined, their size, the type of table (such as clustered, external), etc. Query characteristics include: the list of built-in and custom functions used in the query, the number of subqueries, etc. Finally, applied optimization includes: specific ways to balance joins, perform pruning, parallelize work, compress data, etc.

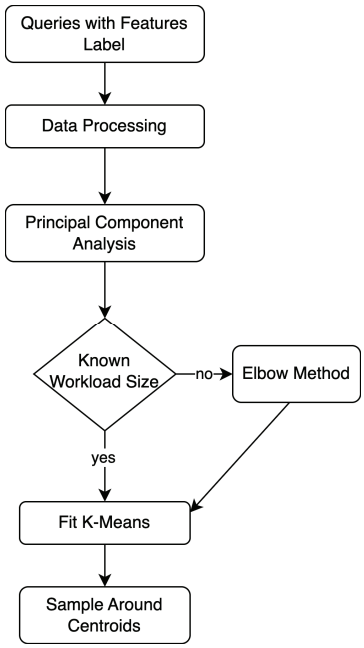

Fig. 1. Procedure flowchart of Snowtrail's new cluatering-based query sampling. The new sampling process generates representative query workloads with high variation in Snowflake features.

The table is dynamically and incrementally updated daily using data already available in our own data warehouse, containing 1-2 billion queries. Most columns are numerical or boolean data, but some columns contain JSON data such as arrays and objects, to support dynamically adding more features without altering the table's schema. For the purpose of vectorizing queries, we select a total of 94 feature columns with strictly numerical or boolean data.

B. Vectorizing and Balancing Queries

After retrieving the query population, we fully vectorize the dataset. We transform boolean feature columns to integers: True to 1, False to 0, and NULL (meaning the feature was not tracked for that specific query) to -1. Lastly, we standardize each value across feature columns by removing the mean and dividing by the standard deviation. Standardization is necessary for the downstream classification task because each feature value range differs greatly. The output is query vectors of 94 dimensions.

C. Sampling with PCA, K-Means, and KDTree

We chose to explore unsupervised learning, rather than supervised learning, for two reasons. Firstly, dimensionality reduction and clustering algorithms generally do not require labeled datasets. Secondly, Snowtrail's preference to dynamically retrieve large sampling populations in real time makes supervision a frequent and expensive job.

Because each query is represented as a 94-dimensional vector, dimensionality reduction is necessary to reduce the loss from the curse of dimensionality [11]. While many techniques are available, we selected Principal Component Analysis (PCA) because it has been shown to produce good results for pre-processing data for many machine learning tasks [12]. PCA is an unsupervised multivariate statistical technique that reduces the number of components in the dataset while optimizing to preserve information by selecting components with the greatest variance [13]. For Snowtrail applications, with default parameters, PCA reduces query features from 94 to 15, which allows variance explained to be consistently preserved above 70%.

K-Means is used for clustering queries and discovering centroids. K-Means is an unsupervised classification algorithm that performs clustering based on minimizing the Euclidean distance between all points and the centroid of the cluster to which they belong [14]. The challenge of K-Means is the NP hard problem of identifying the optimal number of centroids, which we denote by k. Snowtrail allows for the typical search for k (given list of possible values, selecting k based on lowest loss). Given the task of grouping k clusters, K-Means algorithm works at the following:

1) Random selection of k queries as temporary cluster centroids.
2) For each point, calculate Euclidean distance to all centroids and assign the point to the closest centroid to form temporary clusters.
3) Select k new clusters centroids by finding the center of mass of each current cluster.
4) Repeat steps 2 and 3 until clusters are fixed.

K-Means is proved to finitely converge [15], and a special K-Means++ initialization procedure [16] replaces the random initialization for more consistent and better clustering performances. This is mainly to avoid the random initialization placing cluster centroids too closely. We also tested Mini-Batch K-Means [17], a random batch version of K-Means with lower runtime, but we rejected it for Snowtrail as our experiments found that the decrease in runtime was relatively small, while the clustering was significantly worse.

After confirming cluster centroids, a KDTree data structure [18] is used to store and sample queries. The Snowtrail user can specify any number of queries to select per cluster, and the queries will be selected based on the closest Euclidean distances to cluster centroids. Most Snowtrail applications just sample one query per cluster to maximize variation among queries.

IV. Evaluation

We first run K-Means sampling with various population sizes and sample workload sizes to analyze runtime. Then, we conduct test runs to evaluate the performance differences against uniform random sampling, which is Snowtrail's current main sampling method.

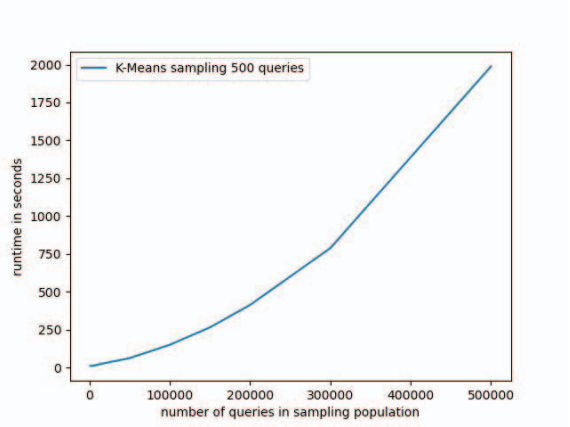

Fig. 2. K-Means is a superpolynomial algorithm that becomes less practical with larger datasets.

A. Runtime Analysis

The running time for the K-Means sampling is dependent on the sampling population, number of clusters, data dimensions, and convergence speed. While research on K-Means runtime upper-bound is ongoing, it is generally believed that K-Means has superpolynomial time complexity [16]. This means that Snowtrail should not use K-Means to sample overly large datasets, provided that a reasonably size random sample provides a close approximation to the distribution of all Snowflake queries [19].

Runtime statistics on local development environment with real-time Snowhouse data retrieval is displayed in Fig. 2, where fixed-sized 500 queries workloads are selected from populations with increasing size. Increasing sample size with a fixed population size is less costly, as K-Means runs in roughly linear time, as shown in Fig. 3. Although not selected for our application, a maximum iteration limit is available to prevent runtime explosion, but at the cost of guaranteed convergence.

B. Workload Comparison with Random Sampling

To simulate performance for realistic Snowtrail runs, the testing parameters are set as follows: the query population is a fixed random selection of 100,000 customer queries in the newest Snowflake version. 15 components are preserved through PCA to preserve explained variance above 70%. Then, we sample with various sample sizes from the query population repeatedly for 10 times for consistency, and take the mean value. At the same time, we generate a random sample with the same population and sample size. The sample sizes are each multiple of 100 from 100 to 2,000. We only compare statistics with components post-PCA processing, and the

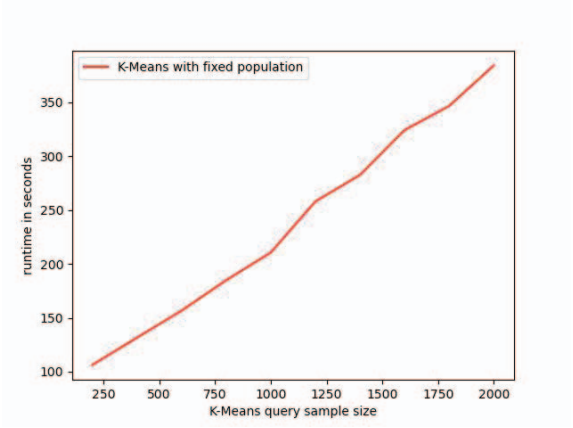

Fig. 3. Runtime increases linearly with sample size. However, common Snowtrail workloads are limited to around 1000 queries.

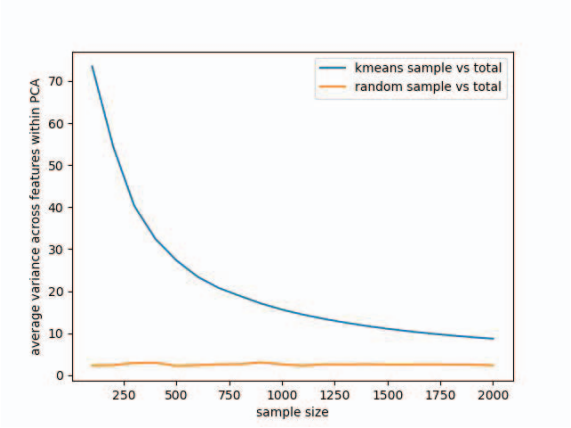

Fig. 4. Column-wise variance measures the spread of queries in one dimension per workload. The mean of variances are taken to represent the spread of the specific workload. K-Means selected samples have much greater spread than randomly selected samples. Randomly selected samples have a similar spread as the population. As sample size increases, the spread of K-Means selected samples converges towards the population's spread.

removed components are assumed to be similarly distributed between the K-Means selected queries and randomly selected queries.

To evaluate the spread of selected queries compared to random sampling, we measure workloads' average variance across feature columns compared to the sampling population. We take the mean of column variances across 10 runs per feature, and take the mean across features. Fig. 4 shows the change in variance as sample size increases. K-Means selected sample has much greater variance, especially when sample size is small, which suggests much greater spread in features' values within the workload. The variance of the randomly selected workload is almost identical vs. the true population (which is 2.47). Notice that the K-Means sample's variance also converges towards the population as sample size grows. This is likely due to K-Means centers also following a version of the Central Limit Theorem [20]. This result is desirable for Snowtrail runs with constraint on workload size, showing that we can generate concise workloads with great spread in terms of Snowflake features.

We use Kullback–Leibler (KL) divergence [21] to measure the similarity in distribution between K-Means samples and random samples. Given two d dimensional samples with dimensional mean matrices μ_1, μ_2 and dimensional covariance matrices Σ_1, Σ_2, a multivariate version of KL divergence is implemented:

$$KL = \frac{1}{2}[log(\frac{|\Sigma_2|}{|\Sigma_1|}) - d + tr(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1)] \quad (1)$$

Results displayed in Fig. 5 indicate that K-Means sample results differ greatly from the population when sample size is small, and converge towards the results for the population. The dissimilarity is because K-Means does not cluster exactly based on population density, and resulting centroids are likely more distributed across values than concentrated in alignment with density. However, as the number of clusters increases, K-

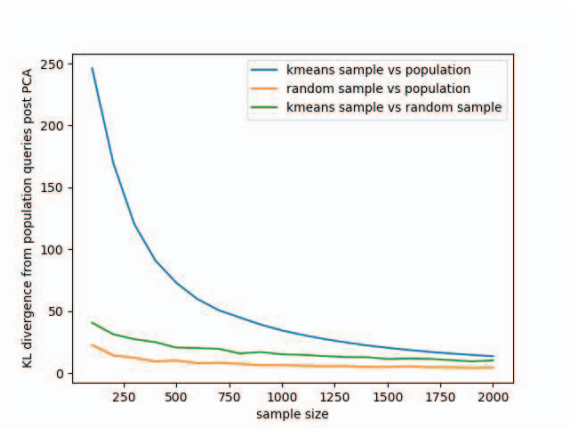

Fig. 5. Kullback–Leibler divergence suggests that the K-Means sample grows increasingly similar the population as the sample size increases. This is because K-Means does not follow the density or distribution of points, but still follows the general Central Limit Theorem.

Means samples are forced to compress in distance, making it more similar to the population. The very large KL divergence at small sample sizes suggests the possible need to better simulate the population, and the inverse logarithmic shaped curve means that the similarity to the population can quickly increase without overly large datasets.

While KL divergence is a general measure of discrepancy between distributions, the Kolmogorov-Smirnov (KS) test is a null hypothesis test that distinguishes between populations [22]. While the multidimensional KS test have been widely researched [23], we will use 2-sample KS tests across individual features and average the results. The test takes in samples d_1, d_2 and outputs test statistic s, where a higher value indicates high likelihood of the two samples being different.

As shown from Fig. 6, the score of the random sample

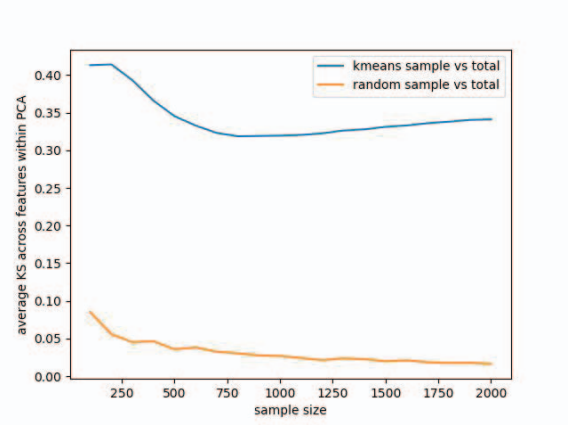

Fig. 6. Kolmogorov-Smirnov test suggests that the K-Means sample is highly distinct from the query population, and the random sample converges to the population as said by the Central Limit Theorem.

converges to zero, again supported by the Central Limit Theorem. However, there is no clear trend of convergence for the K-Means sample. Overall, the KS test strongly suggests the distinction from the population for the K-Means sampling. One possible reason is that K-Means can over-sample low-density extreme examples to minimize loss, yet the KS test takes into account only the maximum difference between distribution curves, which can be skewed by such samples.

V. Automated sample size selection

As suggested by the inverse logarithmic shapes of the KL divergence scores and mean variance across column values, there is an ideal point where the marginal increase in similarity to the population decreases with sample size, while still carrying significantly larger variance among queries. Thus, we implement an elbow method for selecting a workload size. The method is almost identical to the elbow method popularly used in K-Means implementations for selecting the ideal k value [24]. Upon testing, the elbow is consistent for the KL divergence and variance curves, and the elbow values are generally close between the two curves. In general, the elbow-selected workload consistently has the following traits: concise (less than 1000 queries in our tests), varied in queries features, and similar enough to the population. Snowtrail users now have the option to have the algorithm automatically select the ideal workload size by passing in a list of possible sizes. Note that this is an extended usage of the elbow K-Means method, which means that runtime increases roughly linearly with the increase in possible sample sizes.

VI. Conclusion

Along with the increasing popularity in cloud data services, the growth of Snowflake demands high efficiency from its testing infrastructures. As a key component of Snowflake's development and release testing pipeline, Snowtrail gains the capability to sample customer queries with an ideal balance of broad feature coverage and strong resemblance of popular Snowflake usages. To realize this, we begin with vectorized queries that represent feature characteristics and use PCA to reduce dimensions. Then, we use K-Means to classify queries into clusters and sample within clusters. The method can be fine tuned with different population sizes, sample sizes, PCA components, cluster sizes, and other filters. Furthermore, Snowtrail can optionally determine the ideal sample size using an elbow method analogous to the K-Means elbow method. The strength of our sampling procedure is that the resulting workload has great variation among queries, achieving broad feature coverage given a workload size constraint. Interestingly, a concise workload likely differs in distribution from total customer queries, but this is less important than coverage for Snowtrail purposes, and increasing sample size quickly reduces this gap. Because the approach is based on K-Means, the weaknesses are the polynomial runtime of K-Means and the curse of dimensionality in clustering. Overall, the K-Means sampling procedure gives Snowflake another tool in automated, efficient testing, and is a direct improvement from current Snowtrail sampling methods in terms of broad feature coverage. Outside of Snowflake, provided similar feature stores, similar machine learning based approaches can be generally applied to select workloads for software testing.

Acknowledgment

All research data, software, and systems are provided by Snowflake Snowflake Computing. This work is completed while all authors are under the employment of Snowflake Computing.

References

[1] B. Hayes, "Cloud computing," 2008.
[2] W. Al Shehri, "Cloud database database as a service," *International Journal of Database Management Systems*, vol. 5, no. 2, p. 1, 2013.
[3] B. Dageville, T. Cruanes, M. Zukowski, V. Antonov, A. Avanes, J. Bock, J. Claybaugh, D. Engovatov, M. Hentschel, J. Huang *et al.*, "The snowflake elastic data warehouse," in *Proceedings of the 2016 International Conference on Management of Data*, 2016, pp. 215–226.
[4] J. Yan, Q. Jin, S. Jain, S. D. Viglas, and A. Lee, "Snowtrail: Testing with production queries on a cloud database," in *Proceedings of the Workshop on Testing Database Systems*, 2018, pp. 1–6.
[5] D. M. Abdulqader, A. M. Abdulazeez, and D. Q. Zeebaree, "Machine learning supervised algorithms of gene selection: A review," *Machine Learning*, vol. 62, no. 03, pp. 233–244, 2020.
[6] R. Kumar, M. Singh, P. Kumar, and J. Singh, "Crop selection method to maximize crop yield rate using machine learning technique," in *2015 international conference on smart technologies and management for computing, communication, controls, energy and materials (ICSTM)*. IEEE, 2015, pp. 138–145.
[7] P. Mitra, C. Murthy, and S. K. Pal, "Unsupervised feature selection using feature similarity," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 3, pp. 301–312, 2002.
[8] S.-J. Yen and Y.-S. Lee, "Cluster-based under-sampling approaches for imbalanced data distributions," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5718–5727, 2009.
[9] J. Kang, K. R. Ryu, and H.-C. Kwon, "Using cluster-based sampling to select initial training set for active learning in text classification," in *Advances in Knowledge Discovery and Data Mining: 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26-28, 2004. Proceedings 8*. Springer, 2004, pp. 384–388.
[10] S. Jain, B. Howe, J. Yan, and T. Cruanes, "Query2vec: An evaluation of nlp techniques for generalized workload analytics," *arXiv preprint arXiv:1801.05613*, 2018.

[11] D. L. Donoho *et al.*, "High-dimensional data analysis: The curses and blessings of dimensionality," *AMS math challenges lecture*, vol. 1, no. 2000, p. 32, 2000.
[12] G. T. Reddy, M. P. K. Reddy, K. Lakshmanna, R. Kaluri, D. S. Rajput, G. Srivastava, and T. Baker, "Analysis of dimensionality reduction techniques on big data," *Ieee Access*, vol. 8, pp. 54 776–54 788, 2020.
[13] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.
[14] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, 2020.
[15] S. Z. Selim and M. A. Ismail, "K-means-type algorithms: A generalized convergence theorem and characterization of local optimality," *IEEE Transactions on pattern analysis and machine intelligence*, no. 1, pp. 81–87, 1984.
[16] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006.
[17] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 1177–1178.
[18] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.
[19] S. G. Kwak and J. H. Kim, "Central limit theorem: the cornerstone of modern statistics," *Korean journal of anesthesiology*, vol. 70, no. 2, pp. 144–156, 2017.
[20] D. Pollard, "A central limit theorem for *k*-means clustering," *The Annals of Probability*, vol. 10, no. 4, pp. 919–926, 1982.
[21] J. M. Joyce, "Kullback-leibler divergence," in *International encyclopedia of statistical science*. Springer, 2011, pp. 720–722.
[22] V. W. Berger and Y. Zhou, "Kolmogorov–smirnov test: Overview," *Wiley statsref: Statistics reference online*, 2014.
[23] G. Fasano and A. Franceschini, "A multidimensional version of the kolmogorov–smirnov test," *Monthly Notices of the Royal Astronomical Society*, vol. 225, no. 1, pp. 155–170, 1987.
[24] P. Bholowalia and A. Kumar, "Ebk-means: A clustering technique based on elbow method and k-means in wsn," *International Journal of Computer Applications*, vol. 105, no. 9, 2014.