Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

June 2026. Vol. 5. No. 3; e-ISSN: 2808-4519

Application of the K-Means Clustering Algorithm in the Analysis of Popularity and Growth Trends of Python Packages on the PyPI Dataset

Muhammad Rafli Wijaya[1*], M Gali Almahdi[2], Sebastian Saut Marulitua Sinaga[3], Benedict Sandi Pangestu Rosa[4]

[1,2,3,4]*Computer Science, Universitas Negeri Medan*
Jl. Willem Iskandar, Ps. V, Medan Estate, Kec. Percut Sei Tuan, Kab. Deli Serdang, Kota Medan – Sumatera Utara
rafliwijaya.4243250017@mhs.unimed.ac.id[1*], galialmahdi.4243250042@mhs.unimed.ac.id[2],
sebastian.4242550005@mhs.unimed.ac.id[3], benedict.4243250006@mhs.unimed.ac.id[4]

Abstract

The rapid growth of the Python ecosystem has led to an increasing number of packages on the Python Package Index (PyPI), generating a massive volume of download data. This data can be utilized to analyze popularity levels and growth trends of libraries used by the developer community. This study aims to identify popularity patterns and growth trends of Python packages using the K-Means Clustering algorithm. The dataset was obtained from PyPI via the Google BigQuery platform with a one-year observation period using a 1% sampling technique. The pre-processing stage included a filtering process to select the 100 packages with the highest number of downloads and the formation of six main features representing the characteristics of library usage patterns. The data was then normalized using Standard Scaling, while the optimal number of clusters was determined using the Elbow Method and evaluated using the Davies-Bouldin Index (DBI) and Silhouette Score. The results showed that the optimal number of clusters is four, with a DBI value of 0.5534 and a Silhouette Score of 0.5748 (the highest among k = 2-10), representing the categories of ecosystem foundation libraries, medium-popularity libraries, libraries with concentrated download spikes, and libraries with very rapid usage growth. These results indicate that K-Means Clustering is effective for identifying popularity patterns and library growth trends in large-scale PyPI datasets.

Keywords:*Data mining, Google BigQuery, K-Means algorithm, Pypi, Python Package*

1. Introduction

Since its first introduction in 1991, Python has continued to evolve, making it one of the most popular programming languages today [1]. One of the main reasons Python has become so popular besides its syntax simplicity, usage flexibility, and its ability to support various fields such as web development, data science, machine learning, AI, scientific computing, big data, automation, and fintech is the existence of an extensive ecosystem of third-party libraries [2]. These various libraries allow developers to accelerate software development by using pre-existing modules [3]. All of these libraries are organized and distributed through the Python Package Index (PyPI), which serves as the official repository for storing and distributing Python packages that are openly accessible to developers worldwide [4].

As the Python ecosystem advances, the number of accessible packages on PyPI has also experienced very rapid growth. Empirical studies of the Python Package Index (PyPI) ecosystem show that previous analyses reported PyPI had approximately 178,592 packages in 2019, while in the most recent period, the number of projects available in the repository reached more than 627,000 packages by 2025 [3]. This massive growth generates a very large volume of data; however, without structured analysis, this data remains merely a pile of records that are difficult to interpret for decision-making by both developers and service providers. Therefore, this research positions download data as the primary indicator to map the level of usage and popularity of libraries within the software developer community. Consequently, conducting an analysis of package download data becomes essential for understanding library utilization patterns and technological development trends within the Python ecosystem [5]. The growth in the number of packages and the increasing download activity by users on PyPI also generate a massive volume of data that continues to grow over time. This download data can reach millions to billions of records within a specific period, making the analysis of the PyPI dataset a prime example of Big Data concept application [6]. Computing platforms must also be capable of processing large-scale datasets efficiently so that the analysis process can be carried out optimally. One technology that can be utilized is Google BigQuery, a cloud-based data analysis service that enables rapid processing of large datasets without requiring complex computing infrastructure and is capable of performing direct queries against large-scale public datasets without the need for local data downloading [7].

In this study, Google BigQuery is utilized to obtain Python package download data from the PyPI dataset over the last one-year period. Given the massive size of the dataset and BigQuery's quota limitations, the data acquisition process was conducted using a 1% sampling technique of the total download data. Based on these sampling results, a number of packages with the highest download counts were then selected for further analysis by retrieving specific download data. The obtained data was subsequently used as the basis for the process of analyzing the popularity and growth trends of Python packages [4]. To identify patterns of popularity and package growth, this study employs data mining techniques with the K-Means clustering algorithm to discover these popularity and growth patterns. The K-Means clustering algorithm is one of the most commonly used clustering methods in data mining techniques [8]. This algorithm works by partitioning data into several pre-determined groups, based on how close or far the data points are from each other, by calculating the Euclidean distance. The objective of this clustering process is for data within one group to have high similarity and to be significantly different from other groups, so that objects with similar characteristics or traits can be placed in the same group [9].

The utilization of K-Means Clustering in analyzing the popularity and growth trends of Python packages offers several advantages. This method is capable of processing large datasets efficiently, making it suitable for application to PyPI download data, which has a very large volume [11]. Furthermore, the results of the clustering process can provide a more structured overview regarding groups of packages that share similar usage characteristics [12]. In this study, the Python programming language is used to implement the K-Means Clustering algorithm in the data analysis process [13]. Based on this background, this research aims to implement the K-Means Clustering algorithm to group Python packages in the PyPI dataset to identify popularity patterns and growth trends in a structured manner using Google BigQuery services. The results of this research are expected to provide deep insights into libraries with high popularity levels as well as those showing significant growth. Additionally, the study is expected to contribute as a reference for the development of data mining techniques in analyzing large-scale datasets in the future [14].

2. Research Methods

In this study, the dataset used first undergoes a pre-processing stage to prepare the data before the analysis process is conducted. Once the pre-processing is complete, the next stage is the application of the clustering method using the K-Means algorithm, which groups the data into several clusters based on the similarity of data characteristics. The optimal number of clusters is determined using three complementary approaches: the Elbow Method, Davies-Bouldin Index (DBI), and Silhouette Score. To evaluate the quality of the grouping results in depth, an evaluation is performed using DBI and Silhouette Score on the selected final clusters. The stages of this research are visually depicted in the form of a research flowchart shown in Figure 1 below.

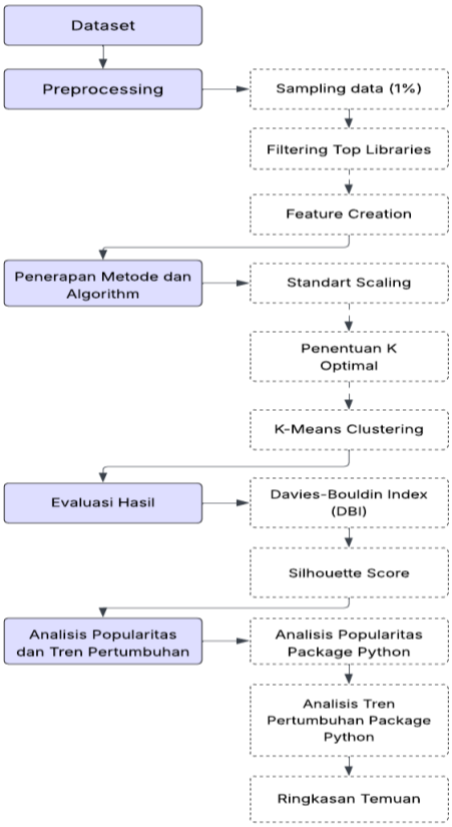

Fig. 1: Research Methods Flow

2.1. Dataset

The dataset used in this study originates from the Python Package Index (PyPI) sourced from the Google BigQuery platform. Download data was retrieved from the public table *bigquery-public-data.pypi.file_downloads*. The observation period is March 1, 2025 – March 07, 2026 (one year). All retrieval and transformation steps were recorded to ensure replicability. This data was selected because it provides information regarding various Python packages along with their download counts, which represent the level of library usage by the global developer community [4].

2.2. Preprocessing

The pre-processing stage is conducted to prepare the data before the analysis process is performed. This process includes data sampling and filtering the top packages/libraries based on the number of downloads to obtain more relevant data. Subsequently, feature creation is carried out to form attributes that represent the characteristics of package usage [15]. To enhance computational efficiency, this study utilizes a 1% sampling technique from the original dataset. This technique is commonly used in big data analysis to reduce processing complexity without losing the primary statistical patterns of the dataset. The download distribution within the PyPI ecosystem is also highly skewed, where a small fraction of libraries has a very high number of downloads while the majority have very low downloads. Therefore, using 1% sampling is still capable of maintaining the representation of popular libraries, which are the focus of this research.

2.2.1. Data Sampling

The data retrieval process was conducted using a 1% systematic sampling technique across all download records in the *file_downloads* table on Google BigQuery. This technique was implemented using the TABLESAMPLE SYSTEM (1 PERCENT) clause, which randomly selects 1% of all data blocks so that the processed data size becomes significantly smaller yet remains statistically representative. The data was then aggregated using GROUP BY project, DATE(timestamp) operations, so that each row in the final result represents the total downloads of a package on a specific date within the one-year observation period.

2.2.2. Filtering Top Libraries

From the sampling results, thousands of Python packages were obtained with widely varying download volumes. The download distribution within the PyPI ecosystem is highly skewed, where a small fraction of libraries dominates the majority of total downloads while thousands of other libraries have very low download frequencies. To ensure the analysis process focuses on relevant libraries with meaningful pattern variations, a filtering process was conducted by selecting the top 100 packages (Top 100) based on the highest total downloads throughout the entire observation period. The selection of the 100-package threshold was based on the consideration that this number is sufficient to produce statistically meaningful clustering while including the libraries that are truly actively used by the global Python developer community.

2.2.3. Features Creation

The feature creation stage aims to transform daily time-series data into a concise representation per package in the form of numerical feature vectors. This is necessary because the K-Means algorithm requires a single row of data per object rather than hundreds of rows per package. From the daily data of the Top 100 packages, six features were extracted as follows:

1. total_downloads, which is the total number of downloads during the entire observation period that represents the overall popularity level.

   $$total\_downloads = \sum_{t=1}^{T} d_t \quad (1)$$

   where *dt* is the number of downloads on day *t.*

2. avg_daily, which is the average daily downloads reflecting usage consistency.

   $$avg\_daily = {}^{1}_{T}\sum_{t=1}^{T} d_t \quad (2)$$

3. std_daily, which is the standard deviation of daily downloads that measures the level of fluctuation or volatility in download patterns.

   $$std\_daily = \sqrt{{}^{1}_{T}\Sigma\,(d_t - \bar{d})^2} \quad (3)$$

4. active_days, which is the total number of days the package is recorded to have downloads, representing the level of activity.
5.

   $$acive\_days = count(d_t) \quad (4)$$

6. max_daily, which is the highest peak downloads in a single day that captures the moment of maximum popularity.

   $$max\_daily = max(d_t) \quad (5)$$

7. growth_rate, which is the ratio comparing the average monthly downloads in the last month to the first month of the observation period, measuring the package's growth trend. A positive growth_rate value indicates the package is experiencing growth, while a negative value indicates a decline in usage trends.

   $$growth\_rate = \frac{avg_{lastmonth} - avg_{firstmonth}}{avg_{firstmonth}} \quad (6)$$

The final result of this stage is a feature matrix of size 100 rows x 6 columns that is ready to be processed in the next stage.

2.3. Application Of Methods And Algorithms

2.3.1. Standart Scaling

Standard Scaling (Z-score normalization) is the method used for data normalization to equalize the scale between features before the clustering process is conducted. Without normalization, features with large value scales such as total_downloads, which can reach hundreds of millions would dominate the Euclidean distance calculation and ignore the contribution of smaller-valued features like growth_rate [16]. Standard Scaling transforms each feature using the formula:

$$z = \frac{(x - \mu)}{\sigma} \tag{7}$$

Where x is the original value, μ is the mean of all values for that feature across 100 packages, and σ is the standard deviation. The transformation results in a distribution with a mean of 0 and a standard deviation of 1 for each feature, ensuring that all features contribute equally to the distance calculation between packages during the clustering process [17].

2.3.2. Determination of Optimal K

Determining the optimal number of clusters is performed using three complementary approaches run simultaneously for each k value from 2 to 10 on the normalized feature matrix (100 rows × 6 features). These three approaches are the Elbow Method, Davies-Bouldin Index (DBI), and Silhouette Score. In each experiment, the inertia value is calculated the total sum of squared distances from each data point to its respective cluster centroid. As the value of k increases, the inertia value decreases because the data is grouped more specifically. The optimal point is chosen at k where the rate of inertia decrease begins to slow significantly, forming an angle resembling an elbow on the graph [18]. All these calculations are executed using Python with the scikit-learn library.

2.3.3. K-Means Clustering

The clustering process is performed using the K-Means algorithm with a cluster count of k=4 on the normalized feature matrix [19]. The algorithm is run 10 times with different centroid initializations (n_init=10) to avoid local optimum solutions, and the best result is selected based on the lowest inertia value. The random_state=42 parameter is used to ensure reproducible results.

2.4. Evaluation of Results

2.4.1. Davies-Bouldin Index (DBI)

The evaluation of clustering quality is conducted, among others, using the Davies-Bouldin Index (DBI), which is calculated using the davies_bouldin_score function from the scikit-learn library in Python. DBI measures clustering quality based on two aspects simultaneously:

1. how compact each cluster is, namely the average distance of data points to their own cluster centroid (intra-cluster distance).
2. how far apart the centroids of different clusters are (inter-cluster distance). DBI is formulated as the average of the maximum ratio (Si + Sj) / Mij for each pair of clusters, where Si and Sj are the average intra-cluster distances of clusters i and j, while Mij is the distance between the centroids of clusters i and j. The smaller the DBI value, the better the clustering quality because the clusters are both compact and well-separated from one another [20].

2.4.2. Silhouette Score

In addition to DBI, the evaluation is also conducted using the Silhouette Score, which measures the quality of grouping from the perspective of each individual data point. For each data point i, the silhouette value s(i) is calculated using the formula:

$$s(i) = \frac{[b(i) - a(i)]}{\max\{a(i), b(i)\}} \tag{8}$$

Where a(i) is the average distance from point i to all other points within the same cluster (intra-cluster distance), and b(i) is the average distance from point i to all points in the nearest neighboring cluster (nearest-cluster distance). The s(i) value ranges from -1 to +1, where a value approaching +1 indicates the data point is in the correct cluster, a value approaching 0 indicates the point is on the border between clusters, and a negative value indicates possible misplacement. The overall Silhouette Score is the average s(i) of all data points and is used in this study to help determine the optimal k alongside the Elbow Method and DBI, and evaluate the quality of the placement of each package into its respective cluster.

2.5. Popularity Analysis And Growth Trends

After the clustering and evaluation processes are completed, the next stage is the interpretation of cluster results to address the two main research objectives: analyzing the popularity and growth trends of Python packages within the PyPI ecosystem. Popularity analysis is conducted based on three primary features *total_downloads, avg_daily*, and *active_days* which together reflect how widely and consistently a library is used by the developer community. Growth trend analysis is performed based on the *growth_rate* feature, which is the ratio comparing the average monthly downloads in the last month to the first month of the observation period. A positive *growth_rate* value indicates a growing library, a value near zero shows a stable trend, and a negative value indicates a decline in usage. Both analyses are conducted interpretatively based on the average feature values per cluster generated by the K-Means algorithm, allowing each cluster to be categorized and its usage pattern characteristics explained contextually within the PyPI ecosystem.

3. Result and Discussion

3.1. Determination of Optimal K

Determining the optimal number of clusters is performed by running the K-Means algorithm for each k value from 2 to 10, then comparing the Inertia, Davies-Bouldin Index (DBI), and Silhouette Score values simultaneously. The complete results of the three metrics for each k value are displayed in the following Table 1.

Table 1: Determination of Optimal K

Nilai K	Inertia	DBI	Silhoutte Score
2	398.1956	1.0125	0.5600
3	233.2073	0.6199	0.5418
4	**142.8125**	**0.5534**	**0.5748**
5	105.7191	0.6764	0.4378
6	79.9842	0.5506	0.4411
7	65.5934	0.6440	0.3759
8	54.3331	0.5772	0.3604
9	45.3269	0.4779	0.3664
10	37.9493	0.5709	0.3490

Table 1: Inertia, DBI, and Silhouette Score values for each K value

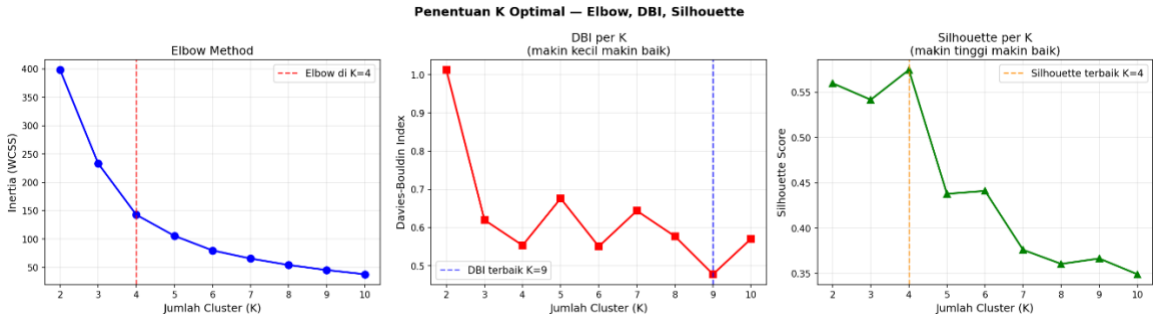

Fig. 2: Elbow Method, DBI and Silhouette per K value

Based on Table 1 and Figure 2, it is evident that the decrease in inertia is very sharp from k=2 (398.20) to k=3 (233.21) with a difference of 164.99, followed by k=3 to k=4 (142.81) with a difference of 90.40. After k=4, the decrease begins to level off significantly; the difference from k=4 to k=5 is only 37.09 and continues to diminish until k=10. The identified elbow point is at k=4, which is the point where adding more clusters no longer provides a proportional reduction in inertia. The DBI value at k=4 of 0.5534 is also considered good (below 1.0), even though the overall lowest DBI is at k=9 (0.4779). More importantly, the Silhouette Score at k=4 of 0.5748 is the highest among all tested k values, being higher than k=2 (0.5600), k=3 (0.5418), and k=5 through k=10, which are all below 0.50. The simultaneous convergence of the three indicators (the Inertia elbow point, a good DBI, and the highest Silhouette Score) at k=4 provides strong confirmation that k=4 is the optimal number of clusters. Using k=9, which has the smallest DBI (0.4779) but a Silhouette Score of only 0.3664 for 100 data points, is deemed too granular and difficult to interpret scientifically. Therefore, k=4 is established as the optimal number of clusters, considering the balance between cluster separation quality and the ease of result interpretation.

3.2. K-Means Clustering Results

The clustering process using the K-Means algorithm with k=4 produced four clusters with varying distributions and characteristics. The algorithm was run 10 times (n_init=10) with varied centroid initializations to avoid local optimum solutions, and the best result was selected based on the lowest inertia value. The following Table 2 summarizes the average values of the six features for each formed cluster.

Klaster	Total Unduh	Avg Daily	Active Days	Max Daily	Growth Rate	Total Project	Kategori
0	34.612.130	699.032	49	2.085.340	+0.24	75	Library Populer Menengah
1	92.235.519	825.198	111	2.917.832	+0.19	20	Library Fondasi Ekosistem
2	30.531.250	1.323.238	23	6.263.035	+3.59	3	Lonjakan Terpusat
3	27.290.160	667.692	41	1.961.234	+84.47	2	Pertumbuhan Ekstrem

Table 2: Features per cluster of K-Means results (k=4)

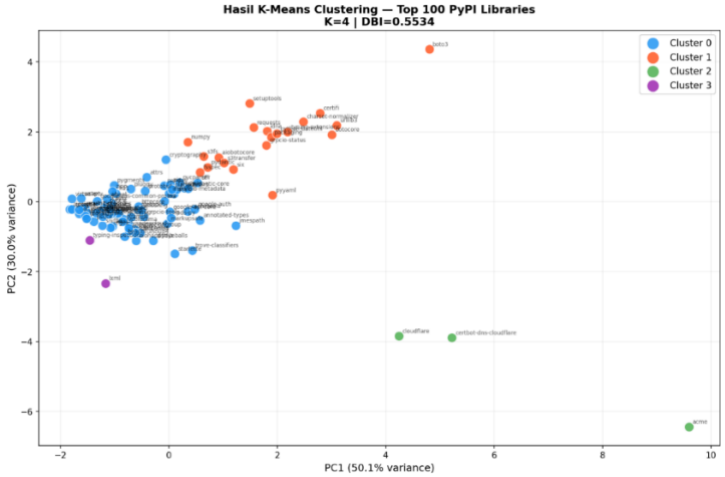

Fig. 3: Results of the distribution of the top 100 libraries

Based on Table 2 and Figure 3, the four clusters exhibit significantly different characteristics from one another. Cluster 1 (20 packages) has the highest average total downloads at 92.2 million with an average of 111 active_days, reflecting libraries that are downloaded almost every day consistently throughout the observation period. This cluster consists of foundational packages in the Python ecosystem such as boto3, urllib3, certifi, botocore, setuptools, requests, numpy, pydantic, and fsspec, which generally serve as core dependencies automatically installed by thousands of other Python projects.

Cluster 0 (75 packages) is the largest group, with an average of 34.6 million total downloads and an average of 49 active_days. This group reflects libraries that are widely used in specific domains but are not as foundational as those in Cluster 1, such as pandas, pytest, cryptography, jinja2, pillow, click, and scipy, which are popular in the fields of data science, security, and web development.

Cluster 2 (3 packages) exhibits a unique pattern: a very high daily download average (1.3 million/day) yet active for only 23 days. The packages in this cluster (acme, certbot-dns-cloudflare, cloudflare) indicate massive downloads concentrated over a short period, most likely due to automated cloud infrastructure deployment processes and SSL certificate tasks run on a scheduled basis.

Cluster 3 (2 packages) is the smallest group but the most prominent in terms of growth, with an average growth_rate of +84.47. This cluster consists of lxml (growth_rate +109.83) and typing-inspection (growth_rate +59.10), two libraries that experienced extreme download surges during the observation period. The surge in typing-inspection was likely triggered by the massive adoption of modern frameworks such as pydantic v2 and mypy, while the surge in lxml is related to the increasing need for XML/HTML parsing in automation and web scraping projects.

3.3. Final Cluster Quality Evaluation

After k=4 was established as the optimal number of clusters and the clustering process was completed, an in-depth evaluation of the quality of the resulting clusters was conducted using two complementary metrics: the Davies-Bouldin Index (DBI) and the Silhouette Score.

3.3.1. Davies-Bouldin Index (DBI) Evaluation

Based on Table 1 evaluation results at k=4, the obtained DBI value is 0.5534. This value falls within the 0.5–1.0 range, which is considered good, indicating that the four formed clusters have an adequate level of compactness and separation with minimal overlap between clusters. It should be noted that the overall smallest DBI value is at k=9 (DBI=0.4779), however, using nine clusters for a dataset of 100 libraries is deemed too granular and complicates scientific interpretation. The DBI result of 0.5534 at k=4 confirms that the six features used are capable of adequately distinguishing the characteristics of download patterns between library groups.

3.3.2. Silhouette Score Evaluation

The overall Silhouette Score for k=4 is 0.5748, which is the highest value compared to all tested k values (k=2 to k=10), confirming that k=4 is the optimal choice. The value of 0.5748 falls within the "good" interpretation range (0.50–0.70), indicating that the majority of libraries have been placed into statistically appropriate clusters. The per-cluster analysis shows the results as presented in the following Table:

Table 3: Silhouette Score per Project

Klaster	Silhouette Score	Total Project
0	0.6197	75
1	0.4484	20
2	0.4505	3
3	0.3412	2

Table 4: Package with the highest Silhouette Score

Package	Silhouette Score	Klaster
scipy	0.7514	0
pathspec	0.7488	0
pillow	0.7453	0
tzdata	0.7441	0
request-authlib	0.7424	0

Cluster 0 has the highest average Silhouette (0.6197) with 75 packages, meaning that mid-tier popular libraries possess the most homogeneous characteristics and clearly defined separation boundaries. The five libraries with the highest Silhouette scores are scipy (0.7514), pathspec (0.7488), pillow (0.7453), tzdata (0.7441), and requests-oauthlib (0.7424), all of which originate from Cluster 0. Conversely, the libraries with the lowest scores are fsspec (0.0014) and cryptography (0.1367), which are located on the boundaries between clusters. Crucially, none of the 100 packages have a negative Silhouette value, meaning that no package was statistically misplaced within its cluster.

3.4. Popularity Analysis and Growth Trends

Based on the results of the K-Means clustering with k=4, which has been evaluated using DBI and the Silhouette Score, an in-depth analysis was conducted on the two primary aspects serving as the research objectives: the popularity and growth trends of Python packages within the PyPI ecosystem. This analysis interprets the characteristics of each cluster contextually, based on the average values of the six extracted features.

3.4.1. Python Package Popularity Analysis

The popularity of a Python package in this study is measured based on three primary features: total_downloads (total number of downloads), avg_daily (average daily downloads), and active_days (number of recorded active days). Together, these three features reflect how extensively and consistently a library is used by the global Python developer community, as shown in Table 2.

Cluster 1 represents the most popular group of packages in the PyPI ecosystem, with an average of 92.2 million total downloads and an average of 111 active days out of the 365-day observation period. The high value of active_days indicates that the libraries in this cluster are downloaded almost every day consistently, rather than just at specific moments. This cluster consists of 20 packages that serve as the core foundation of the Python ecosystem: boto3, urllib3, certifi, botocore, setuptools, charset-normalizer, typing-extensions, idna, requests, python-dateutil, packaging, grpcio-status, aiobotocore, numpy, six, s3transfer, s3fs, pydantic, pyyaml, and fsspec. These packages are generally core dependencies automatically installed by thousands of other Python projects, resulting in very high and stable download volumes.

Cluster 0 is the largest group, containing 75 packages that can be categorized as mid-tier popular libraries. With an average of 34.6 million total downloads and an average of only 49 active_days, this group reflects libraries that are indeed widely used but are not as foundational as those in Cluster 1. Libraries such as pandas, pytest, cryptography, jinja2, pillow, click, and scipy are included here popular within specific domains (data science, web, security) but not necessarily serving as universal dependencies.

Cluster 2 (3 packages) shows a popularity profile that differs from the two previous clusters. Although the total downloads during the observation period are relatively lower (averaging 30.5 million), this cluster actually records the highest average daily downloads among all clusters, at 1.3 million downloads per day. However, this high avg_daily does not reflect broad and evenly distributed popularity, but is instead concentrated within only 23 active days. This indicates that the packages in this clusteracme, certbot-dns-cloudflare, and cloudflareare not used routinely by the general community, but are instead executed massively in short bursts by specific infrastructure automation systems.

Cluster 3 (2 packages) is the group with the fewest members and the lowest absolute popularity level among the four clusters, with an average total downloads of 27.3 million and 41 active_days. Nevertheless, this cluster is actually the most interesting in terms of growth dynamics. The packages in this cluster, lxml and typing-inspection, have not yet reached the stable mass adoption level of Cluster 1, but their extreme growth trajectories indicate that these two libraries are currently in the early phase of a major adoption wave that is likely to continue increasing in the next period.

3.4.2. Table captions

Growth trends are measured using the growth_rate feature, which is the ratio comparing the average monthly downloads of the last month of the observation period to its first month. A positive value indicates that the library is growing, a value near zero suggests a stable trend, and a negative value indicates a decline in usage.

Cluster 3 recorded the most extreme growth trend among all clusters. This cluster consists of only two packages: lxml with a growth_rate of +109.83 and typing-inspection with a growth_rate of +59.10. These values mean that lxml downloads in the last month of observation reached more than 100 times the volume of the first month, while typing-inspection grew by approximately 59 times. The surge in typing-inspection is highly likely triggered by the massive adoption of modern frameworks like pydantic v2 and mypy, which have begun utilizing this library for dynamic data type inspection. Meanwhile, the spike in lxml is likely related to the increasing demand for XML/HTML parsing in automation and web scraping projects.

Cluster 2 also exhibits prominent growth characteristics with an average growth_rate of +3.59, albeit with a different pattern. The packages in this cluster (acme, certbot-dns-cloudflare, cloudflare) maintain very high daily downloads (averaging 1.3 million/day) but are only active for 23 days. This pattern indicates massive downloads concentrated within short periods, most likely as a result of scheduled automated cloud infrastructure deployment processes and SSL certificate management.

On the other hand, some packages in Cluster 0 show a negative trend that warrants closer attention. Several libraries with the lowest growth_rate include platformdirs (−0.69), httpcore (−0.74), markupsafe (−0.74), pyparsing (−0.69), and jmespath (−0.77). This decline does not necessarily mean these libraries are being abandoned; instead, it may indicate a consolidation of usage, a shift toward newer versions or alternatives, or changes in the dependency ecosystem that affect their installation frequency.

Cluster 1, which is the most popular group, actually shows a relatively moderate growth_rate (averaging +0.19). This is reasonable because foundational libraries such as urllib3, certifi, and requests have already reached a very high and stable level of adoption, leaving limited room for further growth. The stable growth in this group indicates the maturity of the ecosystem and a high level of dependency on these specific libraries.

3.5. Summary Of Findings

Overall, the results of the K-Means clustering with k=4 successfully reveal four distinct patterns within the PyPI ecosystem, namely:

1. foundational libraries that dominate in volume and consistency.
2. mid-tier popular libraries serving specific domains.
3. libraries with concentrated downloads due to automation.
4. libraries undergoing exponential growth.

These findings align with the research objectives to analyze the popularity and growth trends of Python packages, proving that the K-Means algorithm with the right features is capable of meaningfully distinguishing library usage patterns within the PyPI ecosystem. The results of this study provide practical insights for software developers in choosing the right libraries tailored to their specific needs.

4. Conclusion

This research successfully implements the K-Means Clustering algorithm to transform large-scale Python package download data from PyPI into structured and interpretable insights. By utilizing Google BigQuery as the data retrieval platform and applying engineering to six download characteristic features, this study is able to identify the popularity patterns and growth trends of Python libraries in a structured manner, fulfilling the primary objectives of this research.

The convergence of three complementary metrics, namely the Elbow Method, Davies-Bouldin Index (DBI=0.5534), and Silhouette Score (0.5748, the highest among k=2–10) simultaneously confirms that k=4 is the optimal cluster configuration. The clustering results reveal four distinct patterns in the PyPI ecosystem:

1. the foundation of the library ecosystem that dominates in terms of download volume and consistency.
2. mid-tier popular libraries serving specific domains.
3. libraries with downloads resulting from cloud infrastructure automation processes.
4. libraries experiencing exponential growth as a signal of a wave of new technology adoption in the Python ecosystem.

These findings demonstrate that K-Means Clustering with appropriate feature engineering is effective for meaningfully distinguishing library usage patterns in the large PyPI dataset.

These findings can provide practical insights for software developers, foundational libraries ensure long-term stability, while libraries with exponential growth can signal early adoption of new technologies. However, this study still has limitations because the features used are limited to download statistics and do not include other factors such as library categories, dependencies between packages, or project metadata. Future research could consider adding semantic features, using other clustering methods for comparison, and expanding the dataset coverage to obtain a more comprehensive picture of the PyPI ecosystem.

References

[1] R. Paramitha and F. Massacci, “Technical leverage analysis in the Python ecosystem,” *Empir. Softw. Eng.*, vol. 28, no. 6, Nov. 2023, doi: 10.1007/s10664-023-10355-2.

[2] S. Raschka, J. Patterson, and C. Nolet, “Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence,” Apr. 01, 2020, *MDPI AG*. doi: 10.3390/info11040193.

[3] J. Mahon, C. Hou, and Z. Yao, “PyPitfall: Dependency Chaos and Software Supply Chain Vulnerabilities in Python,” *ArXiv*, Jul. 2025, [Online]. Available: http://arxiv.org/abs/2507.18075

[4] E. Bommarito and M. Bommarito, “An Empirical Analysis of the Python Package Index (PyPI),” *ArXiv*, Jul. 2019, [Online]. Available: http://arxiv.org/abs/1907.11073

[5] S. Farshidi *et al.*, “Empirical Evaluation of AI-Assisted Software Package Selection: A Knowledge Graph Approach,” *PySelect*, Aug. 2025, [Online]. Available: http://arxiv.org/abs/2508.05693

[6] K. Elwis and H. Hayadi, “FRAMEWORK BIG DATA PADA ANALISIS DAN IMPLEMENTASI PADA PENGOLAHAN DATA SKALA BESAR,” 2025.

[7] B. Berisha, E. Mëziu, and I. Shabani, "Big data analytics in Cloud computing: an overview," *Journal of Cloud Computing*, vol. 11, no. 1, Dec. 2022, doi: 10.1186/s13677-022-00301-w.

[8] E. Xiao, "Comprehensive K-Means Clustering," *Journal of Computer and Communications*, vol. 12, no. 03, pp. 146–159, 2024, doi: 10.4236/jcc.2024.123009.

[9] M. Jillsy Miranda Moningkey, D. Riano Kaparang, and H. Sumual, "The Distribution Pattern of New Students Admissions Using the K-Means Clustering Algorithm," 2024. [Online]. Available: http://ejournal.uksw.edu/ijiteb

[10] D. Galang Ramadhan, I. Prihatini, F. Liantoni, P. Teknik Informatika dan Komputer, and F. Keguruan dan Ilmu Pendidikan, "Analisis Clustering Pengelompokan Penjualan Paket Data Menggunakan Metode K-Means," *Ultimatics : Jurnal Teknik Informatika*, vol. 13, no. 1, p. 33, 2021.

[11] R. Zaib and O. Ourlis, "Large Scale Data Using K-Means," *Mesopotamian Journal of Big Data*, vol. 2023, pp. 36–45, Dec. 2023, doi: 10.58496/MJBD/2023/006.

[12] D. A. Manalu and G. Gunadi, "IMPLEMENTASI METODE DATA MINING K-MEANS CLUSTERING TERHADAP DATA PEMBAYARAN TRANSAKSI MENGGUNAKAN BAHASA PEMROGRAMAN PYTHON PADA CV DIGITAL DIMENSI," *Infotech: Journal of Technology Information*, vol. 8, no. 1, pp. 43–54, Jun. 2022, doi: 10.37365/jti.v8i1.131.

[13] A. Winarta and W. J. Kurniawan, "OPTIMASI CLUSTER K-MEANS MENGGUNAKAN METODE ELBOW PADA DATA PENGGUNA NARKOBA DENGAN PEMROGRAMAN PYTHON," *Jurnal Teknik Informatika Kaputama (JTIK)*, vol. 5, no. 1, 2021.

[14] R. Paramitha, Y. Feng, F. Massacci, and C. E. Budde, "Cross-ecosystem categorization: A manual-curation protocol for the categorization of Java Maven libraries along Python PyPI Topics," *ArXiv*, Mar. 2024, [Online]. Available: http://arxiv.org/abs/2403.06300

[15] I. Patil, D. Makowski, M. S. Ben-Shachar, B. M. Wiernik, E. Bacher, and D. Lüdecke, "datawizard: An R Package for Easy Data Preparation and Statistical Transformations," *J. Open Source Softw.*, vol. 7, no. 78, p. 4684, Oct. 2022, doi: 10.21105/joss.04684.

[16] C. Wongoutong, "The impact of neglecting feature scaling in k-means clustering," *PLoS One*, vol. 19, no. 12, Dec. 2024, doi: 10.1371/journal.pone.0310839.

[17] K. Rinci *et al.*, *BUKU AJAR DATA MINING CV. LUMINARY PRESS INDONESIA*. 2019. [Online]. Available: www.luminarypress.id

[18] C. Shi, B. Wei, S. Wei, W. Wang, H. Liu, and J. Liu, "A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm," *EURASIP J. Wirel. Commun. Netw.*, vol. 2021, no. 1, Dec. 2021, doi: 10.1186/s13638-021-01910-w.

[19] F. Liantoni, *DATA MINING DAN PENERAPAN METODE*. EUREKA MEDIA AKSARA, 2022.

[20] D. Chicco, A. Campagner, A. Spagnolo, D. Ciucci, and G. Jurman, "The Silhouette coefficient and the Davies-Bouldin index are more informative than Dunn index, Calinski-Harabasz index, Shannon entropy, and Gap statistic for unsupervised clustering internal evaluation of two convex clusters," *PeerJ Comput. Sci.*, vol. 11, 2025, doi: 10.7717/peerj-cs.3309.