Click to verify

When trying to analyze data, one approach might be to look for meaningful groups or clusters. Clustering is dividing data into groups based on similarity. And K-means is one of the most commonly used methods in clustering. Why? The main reason is its simplicity. In this tutorial, well start with the theoretical foundations of the K-means algorithm, well discuss how it works and what pitfalls to avoid. Then, well see a practical application of the K-means algorithm with Python using the sklearn library. So, lets get right into it and see what K-means is all about. How Does K-means Clustering Work? Lets say wed like to divide the following points into clusters. First, we must choose how many clusters wed like to have. The K in K-means stands for the number of clusters were trying to identify. In fact, thats where this method gets its name from. We can start by choosing two clusters. The second step is to specify the cluster seeds. A seed is basically a starting cluster centroid. It is chosen at random or is specified by the data scientist based on prior knowledge about the data. One of the clusters will be the green cluster, and the other one - the orange cluster. And these are the seeds. The next step is to assign each point on the graph to a seed. Which is done based on proximity. For instance, this point is closer to the green seed than to the orange one. Therefore, it will belong to the green cluster. This point, on the other hand, is closer to the orange seed, therefore, it will be a part of the orange cluster. In this way, we can assign all points on the graph to a cluster, based on their Euclidean squared distance from the seeds. The final step is to calculate the centroid or the geometrical center of the green points and the orange points. The green seed will move closer to the green points to become their centroid and the orange will do the same for the orange points. From here, we can repeat the last two steps. We can do that 10, 15 or 1000 times until we've reached a clustering solution where we can no longer adjust any of the clusters. Sounds simple, right? However, due to its simplicity, K-means is prone to some issues. What Are the Disadvantages of K-means? One disadvantage arises from the fact that in K-means we have to specify the number of clusters before starting. In fact, this is an issue that a lot of the clustering algorithms share. In the case of K-means if we choose K too small, the cluster centroid will not lie inside the clusters. As we can see, in this example, this is not representative of the data. In cases where K is too large, some of the clusters may be split into two. Reasonable enough, right? Another important issue is that K-means enforces clusters with a spherical shape or blobs. The reason is that we are trying to minimize the distance from the centroids in a straight line. So, if we have clusters, which are more elongated, K-means will have difficulty separating them. Now that youre familiar with what K-means is and how it works, lets bring theory into practice with an example. Whats K-Means Clusterings Application? One of K-means most important applications is dividing a data set into clusters. So, as an example, well see how we can implement K-means in Python. To do that, well use the sklearn library, which contains a number of clustering modules, including one for K-means. Lets say we have our segmentation data in a csv file. After weve read the file (in our case using the pandas method) we can proceed with implementing K-means. Youll need to import KMeans from sklearn cluster. As mentioned above, K-means doesnt tell us how many clusters there are. Instead, it minimizes the Euclidean norm. However, we usually want to be able to determine the number of clusters, too, right? Well, what if we assume there are 2 clusters and we calculate the sum of squared distances within each of the clusters? We can then assume weve got 3 clusters and do the same. Then we can compare the two and see how drastic the change is. In reality, this is exactly what is done. Usually, we run the algorithm, say, 10 different times with 10 different number of clusters. We calculate the Within Cluster Sum of Squares or W C S S for each of the clustering solutions. The WCSS is the sum of the variance between the observations in each cluster. It measures the distance between each observation and the centroid and calculates the squared difference between the two. Hence the name: within cluster sum of squares. So, heres how we use Within Cluster Sum of Squares values to determine the best clustering solution. First, we need to initialize the Within Cluster Sum of Squares variable. It will begin life as an empty list. But there are worse faiths, Im sure. Then well run a for loop, trying out several clustering solutions, calculating their WCSS and adding the value to this list. The next step is determining the size of the loop. How Many Iterations Do We Run? Lets run the code with 10 iterations. So, a range from one to eleven. You can try running the algorithm with a larger number. In this particular example, it wouldnt improve the results, so, well stick with 10. However, depending on the data you may need to increase the number of iterations. In the loop, we run the K-means method. We set the number of clusters to i and initialize with K-means ++. K-means ++ is an algorithm which runs before the actual k-means and finds the best starting points for the centroids. The next item on the agenda is setting a random state. This ensures well get the same initial centroids if we run the code multiple times. Then, we fit the K-means clustering model using our standardized data. The statement fits a K-means clustering model with i clusters to it. And lastly, in each iteration, we add a value to the WCSS array. The WCSS is stored in the inertia attribute. How to Choose the Number of Clusters? The next step is to plot the results. Well plot W C S S values against the number of clusters. Here is the relevant code: On the x-axis, we will have the range from 1 to 11. On the y-axis, we will have the WCSS variable. You can also set a marker and line style for some added visual appeal. Its up to you. And you can see in the picture bellow the result from executing the code: We have our plot, so lets analyze it. We see the function is monotonically decreasing. Sometimes it can be rapidly declining, other times more smoothly. Depending on the shape of this graph, we make a decision about the number of clusters. But how exactly do we choose how many clusters we want? Well use an approach known as the Elbow method. As you can see, this graph looks like an arm with an elbow. The goal here is to spot the elbow itself and take that many clusters. Usually, the part of the graph before the elbow would be steeply declining, while the part after it much smoother. It seems weve got a clear winner: the Elbow on the graph is at the 4-cluster mark. This is the only place until which the graph is steeply declining, while smoothening out afterwards. Note here, that because we have only four clusters, the algorithm with more than 10 will not yield an improvement in the results. But in cases where there are more clusters, a loop with more iterations should be performed. Now we can perform K-means clustering with 4 clusters. We initialize with K-means ++ again and well use the same random state: 42. Finally, we must fit the data. And thats all you need to perform K-means Clustering in Python. Hopefully, we shed some light on how K-means works and how to implement it in Python. You can learn more, check out Customer Analytics in Python course. Then take your skills to the next level with Machine Learning in Python. If youre looking to get familiar with other convenient libraries in Python, you can check out our post on PCA and get to know how to combine PCA and K-means clustering. In case youre new to Python, and youre enthusiastic to learn more, this super-detailed article on learning Python programming will guide you all the way from the installation, through Python IDEs, Libraries, and frameworks, to the best Python career paths and job outlook. Ready to take the next step towards a career in data science? Check out thecomplete Data Science Programtoday. Start with the fundamentals with our Statistics, Maths, and Excel courses, build up step-by-step experience with SQL, Python, R, and Tableau, and upgrade your skillset with Machine Learning, Deep Learning, Credit Risk Modeling, Time Series Analysis, and Customer Analytics in Python. If you still arent sure you want to turn your interest in data science into a solid career, we also offera free preview version of the Data Science Program. Youll receive 12 hours of beginner to advanced content for free. Its a great way to see if the program is right for you. K-Means is an example of a clustering algorithm. Clustering is a fundamental concept in Machine Learning, where the goal is to group a set of objects so that objects in the same group are more similar to each other than to those in other groups. Clustering belongs to the set of unsupervised Machine Learning algorithms, that is no ground truth is needed. Among the various clustering algorithms, K-Means stands out for its simplicity and efficiency. In this blog post, we will explain the algorithm behind K-Means, and see how to implement it in Python.Illustration of clustering.What is K-Means Clustering? - The AlgorithmK-Means clustering is an unsupervised Machine Learning algorithm used to partition an unlabeled, unorganised data set into k clusters, where the number k is defined in advance. Each data point is assigned to the cluster with the closest centroid, based on a predefined distance metric. The algorithm minimizes the inner-cluster variance of the data. More precisely the algorithm follows these steps:1. Initialization: The algorithm starts by selecting k initial centroids. These can be chosen as random points from the data sets or by using more sophisticated methods like K-Means++ to improve convergence. When using the sklearn library, the initial method k-means++ is the default method.2. Assignment Step: For each data point, the distance to each of the centroids is calculated. Each data point is then assigned to the nearest centroid. As distance metric typically the Euclidean distance is chosen, which is also the metric used in the sklearn method. After this step k clusters are formed.3. Update Step: The centroids are recalculated as the mean of all data points assigned to each cluster.Repeat steps 2 and 3 iteratively until the centroids no longer change significantly or the indicated number of iterations is reached. The k clusters after reaching the convergence criterion are the final result.Choosing the Right Number of ClustersOne of the key challenges in K-Means clustering is selecting the optimal number of clusters. Several of methods have been developed to determine the optimal number of clusters. In this post, we will only discuss the most popular ones: the Elbow Method and the Silhouette Score.Elbow MethodThe Elbow Method helps to determine the optimal number of clusters by plotting the sum of squared distances (inertia) from each point to its assigned centroid against the number of clusters. As the number of clusters increases, the inertia decreases. The point where the decrease sharply slows down (forming an elbow) indicates the optimal number of clusters.This methods popularity stems from its simplicity and visual appeal. It allows data scientists and analysts to quickly identify a suitable number of clusters by visual inspection.Pros:Simple to understand and implement.Provides a visual indication of the optimal number of clusters.Cons:The elbow point can sometimes be ambiguous or not very pronounced, making it hard to decide the optimal number of clusters.Silhouette MethodThe Silhouette Score or Silhouette Coefficient evaluates the consistency within clusters and the separation between clusters. It ranges from -1 to 1, with higher values indicating better clustering. The method is in detail described on Wikipedia. The score is calculated as follows:1. step: For each data point i in cluster C_I:Calculate the mean inner-cluster distance (a): This is the average distance between the data point i and all other points in the same cluster.$$a(i) = \frac{1}{|C_I| - 1} \sum_{j\in C_I, jeq i} d(i,j),$$with $|C_I|$ the total number of data points in cluster I, and $d(i, j)$ the distance between data point i and data point j.Calculate the mean nearest-cluster distance (b): This is the average distance between the data point i and all points in the nearest neighboring cluster.$$b(i) = \min_{Jeq I}\frac{1}{C_J}\sum_{j\in C_J} d(i,j),$$where C_J is any other cluster than C_I. Since the minimum over all clusters is taken, this results in the average distance of data point i and all data points in the neighboring cluster. The neighboring cluster is the next best fit cluster for the data point i.2. step: Calculate the silhouette coefficient for the data point i:$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}.$$This equation holds if $|C_I| > 1$, if $|C_I| = 1$, $s(i)$ is defined as 0.The above equation can be rewritten as$$s(i) = \begin{cases}1 - \frac{a(i)}{b(i)} & \text{if } a(i) < b(i) ewline0 & \text{if } a(i) = b(i) ewline\frac{b(i)}{a(i)} - 1 & \text{if } a(i) > b(i)\end{cases}$$s(i) ranges from -1 to 1:s(i) close to 1: The data point is well-clustered. The mean distance to the data points within the cluster is smaller than the mean distance to the data points of the neighboring cluster.s(i) close to 0: The data point is on or very close to the decision boundary between two neighboring clusters.s(i) close to -1: The data point might have been assigned to the wrong cluster. The mean distance to the data points in neighboring cluster is smaller than the mean distance to the data points within the cluster.3. step: Calculate the mean silhouette score for all data points: The overall silhouette score for the clustering is the average silhouette coefficient for all data points.$$S = \frac{1}{N}\sum_{i=1}^N s(i)$$where N is the total number of data points.This method is favored because it provides a more detailed and rigorous evaluation of clustering quality, helping to ensure that the chosen number of clusters truly represents the data structure.Pros:Provides a clear and quantitative measure of clustering quality.Works well for a variety of cluster shapes and densities.Cons:Computationally more intensive than the Elbow Method, especially for large datasets.May not be as intuitive to interpret as the Elbow Method.Other MethodsThe Elbow method and the Silhouette Score are the most popular and the most straight forward methods. There are however several other methods, that are more complex. We are not going to explain them here, but only name a few and link to a reference.Gap StatisticDavies-Bouldin IndexInformation Criterion Approaches (BIC/AIC)Code ExampleWe consider a simple dataset consisting of 28 data points. The data describes the number of followers and the number of posts in a social media application for 28 persons.The data set used in this example.We read this data into a Pandas dataframe and then transform it into a numpy array. 1 2 3 4 5 6 7 8 9101112import pandas as pddata = {"Number of Followers": [0, 10, 15, 20, 5, 38, 20, 15, 210, 200, \\ 220, 190, 0, 6, 15, 10, 200, 180, 150, \\ 170, 8, 12, 210, 230, 190, 220, 210, 190], "Number of Posts": [2, 5, 7, 5, 20, 100, 105, 145, 200, 150, \\ 185, 155, 120, 150, 170, 170, 500, 550, 400, \\ 470, 8, 10, 160, 180, 165, 470, 490, 450] }df = pd.DataFrame(data)X = df[["Number of Followers", "Number of Posts"]].valuesWe use the sklearn method KMeans with the default setup. For this demonstration, the only parameter we change is the number of clusters (n_clusters). We can, however, e.g. change the initialization method (init), the maximal number of iterations (max_iter), and the tolerance (tol) that defines the convergence of the algorithm. For all details, please refer to the documentation.We use the defined data set to fit the KMeans method. We fit the data using 2, 4, and 8 clusters and plot the results. We can then access the calculated centroids using kmeans.cluster_centers_. 1 2 3 4 5 6 7 8 910111213141516171819from sklearn.cluster import KMeansimport matplotlib.pyplot as pltrandom_state=42n_clusters_list = [2, 4, 8]fig, axs = plt.subplots(1, len(n_clusters_list), figsize=(11, 4))for j, n_clusters in enumerate(n_clusters_list): kmeans = KMeans(n_clusters=n_clusters, random_state=random_state) kmeans.fit(X) centers = kmeans.cluster_centers_ axs[j].scatter(X[:, 0], X[:, 1], s=20, c=kmeans.labels_) axs[j].scatter(centers[:, 0], centers[:, 1], c="r", s=40) axs[j].set_title(f"{n_clusters} clusters") axs[j].set_xlabel("Number of Followers") axs[j].set_ylabel("Number of Posts")Clustering the example data with 2, 4, and 8 clusters.In this simple example, we can already visually see that 4 clusters are the correct number, but lets use the above discussed methods to verify this.Elbow MethodTo illustrate the Elbow Method, we fit the data for an increasing number of clusters ranging from 1 to 11, save the inertias (sum of square distances), and plot the result.1234567from sklearn.cluster import KMeansinertia = []K = range(1, 11)for k in K: kmeans = KMeans(n_clusters=k, random_state=0).fit(X) inertia.append(kmeans.inertia_)Elbow curve for the example data.In this plot, we see that at the beginning the curve is falling steeply and this decrease reduces strongly after reaching 4 clusters. This indicates that this is the best number of clusters for this task.Silhouette ScoreThe silhouette score is also a method available in sklearn. In this case, we fit the KMeans method to a range of 2 to 11 clusters. For each fit, we calculate the silhouette score and plot the result.123456789from sklearn.metrics import silhouette_scorefrom sklearn.cluster import KMeanssilhouette_scores = []K = range(2, 11)for k in K: kmeans = KMeans(n_clusters=k, random_state=0).fit(X) score = silhouette_score(X, kmeans.labels_) silhouette_scores.append(score)Silhouette Score for the example data.This plot shows a maximum for 4 clusters, that is the silhouette score is highest for this number of clusters, which indicates the optimal number of clusters.KMeans is an unsupervised machine learning algorithm used for clustering data into k distinct groups based on feature similarity. It works by iteratively assigning data points to the nearest cluster center and then updating the cluster centers to be the mean of the assigned points. The process continues until the cluster assignments no longer change significantly or the maximum number of iterations is reached. It is a popular algorithm due to its simplicity, efficiency with large datasets, and ease of implementation and interpretation. An example of applying KMeans to a more real-world dataset than the simplifies example above can be found on kaggle.If this blog is useful for you, please consider supporting.Data ScienceMachine LearningClusteringunsupervisedVector quantization algorithm minimizing the sum of squared deviationsNot to be confused with k-nearest neighbors algorithm.Part of a series onMachine learningand data miningParadigmsSupervised learningUnsupervised learningSemi-supervised learningSelf-supervised learningReinforcement learningMeta-learningOnline learningBatch learningCurriculum learningRule-based learningNeuro-symbolic AINeuromorphic engineeringQuantum machine learningProblemsClassificationGenerative modelingRegressionClusteringDimensionality reductionDensity estimationAnomaly detectionData cleaningAutoMLAssociation rulesSemantic analysisStructured predictionFeature engineeringFeature learningLearning to rankGrammar inductionOntology learningMultimodal learningSupervised learning(classification regression) ApprenticeshipDecision treesEnsemblesBaggingBoostingRandom forestk-NNLinear regressionNaive BayesArtificial neural networksLogistic regressionPerceptronRelevance vector machine (RVM)Support vector machine (SVM)ClusteringBIRCHCUREHierarchicalk-meansFuzzyExpectationmaximization (EM)DBSCANOPTICSMean shiftDimensionality reductionFactor analysisCCAICALDANMFPCAPGDt-SNESDLStructured predictionGraphical modelsBayes netConditional random fieldHidden MarkovAnomaly detectionRANSACk-NNLocal outlier factorIsolation forestNeural networksAutoencoderDeep learningFeedforward neural networkRecurrent neural networkLSTMGRUESNreservoir computingBoltzmann machineRestrictedGANDiffusion modelSOMConvolutional neural networkU-NetLeNetAlexNetDeepDreamNeural fieldNeural radiance fieldPhysics-informed neural networksTransformerVisionMambaSpiking neural networkMemtransistorElectrochemical RAM (ECRAM)Reinforcement learningQ-learningPolicy gradientSARSATemporal difference (TD)Multi-agentSelf-playLearning with humansActive learningCrowdsourcingHuman-in-the-loopMechanistic interpretabilityRLHFModel diagnosticsCoefficient of determinationConfusion matrixLearning curveROC curveMathematical foundationsKernel machinesBiasvariance tradeoffComputational learning theoryEmpirical risk minimizationOccam learningPAC learningStatistical learningVC theoryTopological deep learningJournals and conferencesAAAIECML PKDDNeurIPSICMLICLRIJCAIMLJMLRRelated articlesGlossary of artificial intelligenceList of datasets for machine-learning researchList of datasets in computer vision and image processingOutline of machine learningvtek-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. k-means clustering minimizes within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimizes squared errors, whereas only the geometric median minimizes Euclidean distances. For instance, better Euclidean solutions can be found using k-medians and k-medoids.The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum. These are usually similar to the expectationmaximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both k-means and Gaussian mixture modeling. They both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the Gaussian mixture model allows clusters to have different shapes.The unsupervised k-means algorithm has a loose relationship to the k-nearest neighbor classifier, a popular supervised machine learning technique for classification that is often confused with k-means due to the name. Applying the 1-nearest neighbor classifier to the cluster centers obtained by k-means classifies new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.Given a set of observations (x1, x2, ..., xn), where each observation is a d {\displaystyle d} -dimensional real vector, k-means clustering aims to partition the n observations into k (n) sets S = {S1, S2, ..., Sk} so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find: a r g m i n S i = 1 k x S i x i 2 = a r g m i n S i = 1 k | S i | Var S i {\displaystyle \mathop {\operatorname {arg\,min} } _{\mathbf {S} }\sum _{i=1}^{k}\sum _{\mathbf {x} \in S_{i}}\left\|\mathbf {x} -{\boldsymbol {\mu }}_{i}\right\|^{2}=\mathop {\operatorname {arg\,min} } _{\mathbf {S} }\sum _{i=1}^{k}|S_{i}|\operatorname {Var} S_{i}} where i is the mean (also called centroid) of points in S i {\displaystyle S_{i}} , i.e. i = 1 | S i | x S i x , {\displaystyle {\boldsymbol {\mu _{i}}}={\frac {1}{|S_{i}|}}\sum _{\mathbf {x} \in S_{i}}\mathbf {x} ,} | S i | {\displaystyle |S_{i}|} is the size of S i {\displaystyle S_{i}} , and {\displaystyle \|\cdot \|} is the usual L2 norm . This is equivalent to minimizing the pairwise squared deviations of points in the same cluster: a r g m i n S i = 1 k 1 | S i | x , y S i x y 2 {\displaystyle \mathop {\operatorname {arg\,min} } _{\mathbf {S} }\sum _{i=1}^{k}\,{\frac {1}{|S_{i}|}}\,\sum _{\mathbf {x} ,\mathbf {y} \in S_{i}}\left\|\mathbf {x} -\mathbf {y} \right\|^{2}} The equivalence can be deduced from identity | S i | x S i x i 2 = 1 2 x , y S i x y 2 {\textstyle |S_{i}|\sum _{\mathbf {x} \in S_{i}}\left\|\mathbf {x} -{\boldsymbol {\mu }}_{i}\right\|^{2}={\frac {1}{2}}\sum _{\mathbf {x} ,\mathbf {y} \in S_{i}}\left\|\mathbf {x} -\mathbf {y} \right\|^{2}} . Since the total variance is constant, this is equivalent to maximizing the sum of squared deviations between points in different clusters (between-cluster sum of squares, BCSS).[1] This deterministic relationship is also related to the law of total variance in probability theory.The term "k-means" was first used by James MacQueen in 1967,[2] though the idea goes back to Hugo Steinhaus in 1956.[3] The standard algorithm was first proposed by Stuart Lloyd of Bell Labs in 1957 as a technique for pulse-code modulation, although it was not published as a journal article until 1982.[4] In 1965, Edward W. Forgy published essentially the same method, which is why it is sometimes referred to as the LloydForgy algorithm.[5]Convergence of k-meansThe most common algorithm uses an iterative refinement technique. Due to its ubiquity, it is often called "the k-means algorithm"; it is also referred to as Lloyd's algorithm, particularly in the computer science community. It is sometimes also referred to as "nave k-means", because there exist much faster alternatives.[6]Given an initial set of k means m1(1), ..., mk(1) (see below), the algorithm proceeds by alternating between two steps:[7]Assignment step: Assign each observation to the cluster with the nearest mean: that with the least squared Euclidean distance.[8] (Mathematically, this means partitioning the observations according to the Voronoi diagram generated by the means.) S i (t) = { x p : x p m i (t) 2 x p m j (t) 2 j , 1 j k } , {\displaystyle S_{i}^{(t)}=\left\{x_{p}:\left\|x_{p}-m_{i}^{(t)}\right\|^{2}\leq \left\|x_{p}-m_{j}^{(t)}\right\|^{2}\ \forall j,1\leq j\leq k\right\},} where each x p {\displaystyle x_{p}} is assigned to exactly one S (t) {\displaystyle S^{(t)}} , even if it could be assigned to two or more of them.Update step: Recalculate means (centroids) for observations assigned to each cluster. m i (t + 1) = 1 | S i (t) | x j S i (t) x j {\displaystyle m_{i}^{(t+1)}={\frac {1}{\left|S_{i}^{(t)}\right|}}\sum _{x_{j}\in S_{i}^{(t)}}x_{j}} The objective function in k-means is the WCSS (within cluster sum of squares). After each iteration, the WCSS decreases and so we have a nonnegative monotonically decreasing sequence. This guarantees that the k-means always converges, but not necessarily to the global optimum.The algorithm has converged when the assignments no longer change or equivalently, when the WCSS has become stable. The algorithm is not guaranteed to find the optimum.[9]The algorithm is often presented as assigning objects to the nearest cluster by distance. Using a different distance function other than (squared) Euclidean distance may prevent the algorithm from converging. Various modifications of k-means such as spherical k-means and k-medoids have been proposed to allow using other distance measures.PseudocodeThe below pseudocode outlines the implementation of the standard k-means clustering algorithm. Initialization of centroids, distance metric between points and centroids, and the calculation of new centroids are design choices and will vary with different implementations. In this example pseudocode, argmin is used to find the index of the minimum value.def k_means_cluster(k, points): # Initialization: choose k centroids (Forgy, Random Partition, etc.) centroids = [c1, c2, ..., ck] # Initialize clusters list clusters = [[] for _ in range(k)] # Loop until convergence converged = false while not converged: # Clear previous clusters clusters = [[] for _ in range(k)] # Assign each point to the "closest" centroid for point in points: distances_to_each_centroid = [distance(point, centroid) for centroid in centroids] cluster_assignment = argmin(distances_to_each_centroid) clusters[cluster_assignment].append(point) # Calculate new centroids # (the standard implementation uses the mean of all points in a # cluster to determine the new centroid) new_centroids = [calculate_centroid(cluster) for cluster in clusters] converged = (new_centroids == centroids) centroids = new_centroids if converged: return clustersCommonly used initialization methods are Forgy and Random Partition.[10] The Forgy method randomly chooses k observations from the dataset and uses these as the initial means. The Random Partition method first randomly assigns a cluster to each observation and then proceeds to the update step, thus computing the initial mean to be the centroid of the cluster's randomly assigned points. The Forgy method tends to spread the initial means out, while Random Partition places all of them close to the center of the data set. According to Hamerly et al.,[10] the Random Partition method is generally preferable for algorithms such as the k-harmonic means and fuzzy k-means. For expectation maximization and standard k-means algorithms, the Forgy method of initialization is preferable. A comprehensive study by Celebi et al.,[11] however, found that popular initialization methods such as Forgy, Random Partition, and Maximin often perform poorly, whereas Bradley and Fayyad's approach[12] performs "consistently" in "the best group" and k-means++ performs "generally well".Demonstration of the standard algorithm1. k initial "means" (in this case k=3) are randomly generated within the data domain (shown in color).2. k clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.3. The centroid of each of the k clusters becomes the new mean.4. Steps 2 and 3 are repeated until convergence has been reached.The algorithm does not guarantee convergence to the global optimum. The result may depend on the initial clusters. As the algorithm is usually fast, it is common to run it multiple times with different starting conditions. However, worst-case performance can be slow: in particular certain point sets, even in two dimensions, converge in exponential time, that is 2(n).[13] These point sets do not seem to arise in practice: this is corroborated by the fact that the smoothed running time of k-means is polynomial.[14]The "assignment" step is referred to as the "expectation step", while the "update step" is a maximization step, making this algorithm a variant of the generalized expectationmaximization algorithm.Finding the optimal solution to the k-means clustering problem for observations in d dimensions is:NP-hard in general Euclidean space (of d dimensions) even for two clusters,[15][16]NP-hard for a general number of clusters k even in the plane,[17]if k and d (the dimension) are fixed, the problem can be exactly solved in time O (n d k + 1) {\displaystyle O(n^{dk+1})} , where n is the number of entities to be clustered.[18]Thus, a variety of heuristic algorithms such as Lloyd's algorithm given above are generally used.The running time of Lloyd's algorithm (and most variants) is O (n k d i) {\displaystyle O(nkdi)} ,[9][19] where:n is the number of d-dimensional vectors (to be clustered)k the number of clustersi the number of iterations needed until convergence.On data that does have a clustering structure, the number of iterations until convergence is often small, and results only improve slightly after the first dozen iterations. Lloyd's algorithm is therefore often considered to be of "linear" complexity in practice, although it is in the worst case superpolynomial when performed until convergence.[20]In the worst-case, Lloyd's algorithm needs i = 2 (n) {\displaystyle i=2^{\Omega ({\sqrt {n}})}} iterations, so that the worst-case complexity of Lloyd's algorithm is superpolynomial.[20]Lloyd's k-means algorithm has polynomial smoothed running time. It is shown that[14] for arbitrary set of n points in [0 , 1] d {\displaystyle [0,1]^{d}} , if each point is independently perturbed by a normal distribution with mean 0 and variance 2 {\displaystyle \sigma ^{2}} , then the expected running time of k-means algorithm is bounded by O (n 34 k 34 d 8 log 4 (n) / 6) {\displaystyle O(n^{34}k^{34}d^{8}\log ^{4}(n)/\sigma ^{6})} , which is a polynomial in n, k, d and 1 / {\displaystyle 1/\sigma } .Better bounds are proven for simple cases. For example, it is shown that the running time of k-means algorithm is bounded by O (d n 4 M 2) {\displaystyle O(dn^{4}M^{2})} for n points in an integer lattice { 1 , , M } d {\displaystyle \{1,\dots ,M\}^{d}} .[21]Lloyd's algorithm is the standard approach for this problem. However, it spends a lot of processing time computing the distances between each of the k cluster centers and the n data points. Since points usually stay in the same clusters after a few iterations, much of this work is unnecessary, making the nave implementation very inefficient. Some implementations use caching and the triangle inequality in order to create bounds and accelerate Lloyd's algorithm.[22][9][23][24][25][26]Main article: Determining the number of clusters in a data setFinding the optimal number of clusters (k) for k-means clustering is a crucial step to ensure that the clustering results are meaningful and useful.[27] Several techniques are available to determine a suitable number of clusters. Here are some of commonly used methods:Elbow method (clustering): This method involves plotting the explained variation as a function of the number of clusters, and picking the elbow of the curve as the number of clusters to use.[28] However, the notion of an "elbow" is not well-defined and this is known to be unreliable.[29]Silhouette (clustering): Silhouette analysis measures the quality of clustering and provides an insight into the separation distance between the resulting clusters.[30] A higher silhouette score indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.Gap statistic: The Gap Statistic compares the total within intra-cluster variation for different values of k with their expected values under null reference distribution of the data.[31] The optimal k is the value that yields the largest gap statistic.DaviesBouldin index: The Davies-Bouldin index is a measure of the how much separation there is between clusters.[32] Lower values of the Davies-Bouldin index indicate a model with better separation.Calinski-Harabasz index: This Index evaluates clusters based on their compactness and separation. The index is calculated using the ratio of between-cluster variance to within-cluster variance, with higher values indicate better-defined clusters.[33]Rand index: It calculates the proportion of agreement between the two clusters, considering both the pairs of elements that are correctly assigned to the same or different clusters.[34] Higher values indicate greater similarity and better clustering quality. To provide a more accurate measure, the Adjusted Rand Index (ARI), introduced by Hubert and Arabie in 1985, corrects the Rand Index by adjusting for the expected similarity of all pairings due to chance.[35]Jenks natural breaks optimization: k-means applied to univariate datak-medians clustering uses the median in each dimension instead of the mean, and this way minimizes L 1 {\displaystyle L_{1}} norm (Taxicab geometry).k-medoids (also: Partitioning Around Medoids, PAM) uses the medoid instead of the mean, and this way minimizes the sum of distances for arbitrary distance functions.Fuzzy C-Means Clustering is a soft version of k-means, where each data point has a fuzzy degree of belonging to each cluster.Gaussian mixture models trained with expectationmaximization algorithm (EM algorithm) maintains probabilistic assignments to clusters, instead of deterministic assignments, and multivariate Gaussian distributions instead of means.k-means++ chooses initial centers in a way that gives a provable upper bound on the WCSS objective.The filtering algorithm uses k-d trees to speed up each k-means step.[36]Some methods attempt to speed up each k-means step using the triangle inequality.[23][24][25][37][26]Escape local optima by swapping points between clusters.[9]The Spherical k-means clustering algorithm is suitable for textual data.[38]Hierarchical variants such as Bisecting k-means,[39] X-means clustering[40] and G-means clustering[41] repeatedly split clusters to build a hierarchy, and can also try to automatically determine the optimal number of clusters in a dataset.Internal cluster evaluation measures such as cluster silhouette can be helpful at determining the number of clusters.Minkowski weighted k-means automatically calculates cluster specific feature weights, supporting the intuitive idea that a feature may have different degrees of relevance at different features.[42] These weights can also be used to re-scale a given data set, increasing the likelihood of a cluster validity index to be optimized at the expected number of clusters.[43]Mini-batch k-means: k-means variation using "mini batch" samples for data sets that do not fit into memory.[44]Otsu's methodHartigan and Wong's method[9] provides a variation of k-means algorithm which progresses towards a local minimum of the minimum sum-of-squares problem with different solution updates. The method is a local search that iteratively attempts to relocate a sample into a different cluster as long as this process improves the objective function. When no sample can be relocated into a different cluster with an improvement of the objective, the method stops (in a local minimum). In a similar way as the classical k-means, the approach remains a heuristic since it does not necessarily guarantee that the final solution is globally optimum.Let (S j) {\displaystyle \varphi (S_{j})} be the individual cost of S j {\displaystyle S_{j}} defined by x S j (x j) 2 {\textstyle \sum _{x\in S_{j}}(x-\mu _{j})^{2}} , with j {\displaystyle \mu _{j}} the center of the cluster.Assignment stepHartigan and Wong's method starts by partitioning the points into random clusters { S j } j { 1 , k } {\displaystyle \{S_{j}\}_{j\in \{1,\cdots k\}}} .Update stepNext it determines the n , m { 1 , , k } {\displaystyle n,m\in \{1,\ldots ,k\}} and x S n {\displaystyle x\in S_{n}} for which the following function reaches a maximum (m , n , x) = (S n) + (S m) (S n { x }) (S m { x }) . {\displaystyle \Delta (m,n,x)=\varphi (S_{n})+\varphi (S_{m})-\varphi (S_{n}\setminus \{x\})-\varphi (S_{m}\cup \{x\}).} For the x , n , m {\displaystyle x,n,m} that reach this maximum, x {\displaystyle x} moves from the cluster S n {\displaystyle S_{n}} to the cluster S m {\displaystyle S_{m}} .TerminationThe algorithm terminates once (m , n , x) {\displaystyle \Delta (m,n,x)} is less than zero for all x , n , m {\displaystyle x,n,m} .Different move acceptance strategies can be used. In a first-improvement strategy, any improving relocation can be applied, whereas in a best-improvement strategy, all possible relocations are iteratively tested and only the best is applied at each iteration. The former approach favors speed, whether the latter approach generally favors solution quality at the expense of additional computational time. The function {\displaystyle \Delta } used to calculate the result of a relocation can also be efficiently evaluated by using equality[45] (x , n , m) = S n S n 1 n x 2 S m S m + 1 m x 2 . {\displaystyle \Delta (x,n,m)={\frac {\mid S_{n}\mid }{\mid S_{n}\mid -1}}\cdot \lVert \mu _{n}-x\rVert ^{2}-{\frac {\mid S_{m}\mid }{\mid S_{m}\mid +1}}\cdot \lVert \mu _{m}-x\rVert ^{2}.} The classical k-means algorithm and its variations are known to only converge to local minima of the minimum-sum-of-squares clustering problem defined as a r g m i n S i = 1 k x S i x i 2 . {\displaystyle \mathop {\operatorname {arg\,min} } _{\mathbf {S} }\sum _{i=1}^{k}\sum _{\mathbf {x} \in S_{i}}\left\|\mathbf {x} -{\boldsymbol {\mu }}_{i}\right\|^{2}.} Many studies have attempted to improve the convergence behavior of the algorithm and maximize the chances of attaining the global optimum (or at least, local minima of better quality). Initialization and restart techniques discussed in the previous sections are one alternative to find better solutions. More recently, global optimization algorithms based on branch-and-bound and semidefinite programming have produced provenly optimal solutions for datasets with up to 4,177 entities and 20,531 features.[46] As expected, due to the NP-hardness of the subjacent optimization problem, the computational time of optimal algorithms for k-means quickly increases beyond this size. Optimal solutions for small- and medium-scale still remain valuable as a benchmark tool, to evaluate the quality of other heuristics. To find high-quality local minima within a controlled computational time but without optimality guarantees, other works have explored metaheuristics and other global optimization techniques, e.g., based on incremental approaches and convex optimization,[47] random swaps[48] (i.e., iterated local search), variable neighborhood search[49] and genetic algorithms.[50][51] It is indeed known that finding better local minima of the minimum sum-of-squares clustering problem can make the difference between failure and success to recover cluster structures in feature spaces of high dimension.[51]A typical example of the k-means convergence to a local minimum. In this example, the result of k-means clustering (the right figure) contradicts the obvious cluster structure of the data set. The small circles are the data points, the four ray stars are the centroids (means). The initial configuration is on the left figure. The algorithm converges after five iterations presented on the figures, from the left to the right. The illustration was prepared with the Mirkes Java applet.[52]k-means clustering result for the Iris flower data set and actual species visualized using ELKI. Cluster means are marked using larger, semi-transparent symbols.k-means clustering vs. EM clustering on an artificial dataset ("mouse"). The tendency of k-means to produce equal-sized clusters leads to bad results here, while EM benefits from the Gaussian distributions with different radius present in the data set.Three key features of k-means that make it efficient are often regarded as its biggest drawbacks:Euclidean distance is used as a metric and variance is used as a measure of cluster scatter.The number of clusters k is an input parameter: an inappropriate choice of k may yield poor results. That is why, when performing k-means, it is important to run diagnostic checks for determining the number of clusters in the data set.Convergence to a local minimum may produce counterintuitive ("wrong") results (see example in Fig.).A key limitation of k-means is its cluster model. The concept is based on spherical clusters that are separable so that the mean converges towards the cluster center. The clusters are expected to be of similar size, so that the assignment to the nearest cluster center is the correct assignment. When for example applying k-means with a value of k = 3 {\displaystyle k=3} onto the well-known Iris flower data set, the result often fails to separate the three Iris species contained in the data set. With k = 2 {\displaystyle k=2} , the two visible clusters (one containing two species) will be discovered, whereas with k = 3 {\displaystyle k=3} one of the two clusters will be split into two even parts. In fact, k = 2 {\displaystyle k=2} is more appropriate for this data set, despite the data set's containing 3 classes. As with any other clustering algorithm, the k-means result makes assumptions that the data satisfy certain criteria. It works well on some data sets, and fails on others.The result of k-means can be seen as the Voronoi cells of the cluster means. Since data is split halfway between cluster means, this can lead to suboptimal splits as can be seen in the "mouse" example. The Gaussian models used by the expectationmaximization algorithm (arguably a generalization of k-means) are more flexible by having both variances and covariances. The EM result is thus able to accommodate clusters of variable size much better than k-means as well as correlated clusters (not in this example). In counterpart, EM requires the optimization of a larger number of free parameters and poses some methodological issues due to vanishing clusters or badly-conditioned covariance matrices. k-means is closely related to nonparametric Bayesian modeling.[53]k-means clustering is rather easy to apply to even large data sets, particularly when using heuristics such as Lloyd's algorithm. It has been successfully used in market segmentation, computer vision, and astronomy among many other domains. It often is used as a preprocessing step for other algorithms, for example to find a starting configuration.Main article: Vector quantizationVector quantization, a technique commonly used in signal processing and computer graphics, involves reducing the color palette of an image to a fixed number of colors, known as k. One popular method for achieving vector quantization is through k-means clustering. In this process, k-means is applied to the color space of an image to partition it into k clusters, with each cluster representing a distinct color in the image. This technique is particularly useful in image segmentation tasks, where it helps identify and group similar colors together.Example image with only red and green channel (for illustration purposes)Vector quantization of colors present in the image above into Voronoi cells using k-meansExample: In the field of computer graphics, k-means clustering is often employed for color quantization in image compression. By reducing the number of colors used to represent an image, file sizes can be significantly reduced without significant loss of visual quality. For instance, consider an image with millions of colors. By applying k-means clustering with k set to a smaller number, the image can be represented using a more limited color palette, resulting in a compressed version that consumes less storage space and bandwidth. Other uses of vector quantization include non-random sampling, as k-means can easily be used to choose k different but prototypical objects from a large data set for further analysis.Main article: Cluster analysisCluster analysis, a fundamental task in data mining and machine learning, involves grouping a set of data points into clusters based on their similarity. k-means clustering is a popular algorithm used for partitioning data into k clusters, where each cluster is represented by its centroid.However, the pure k-means algorithm is not very flexible, and as such is of limited use (except for when vector quantization as above is actually the desired use case). In particular, the parameter k is known to be hard to choose (as discussed above) when not given by external constraints. Another limitation is that it cannot be used with arbitrary distance functions or on non-numerical data. For these use cases, many other algorithms are superior.Example: In marketing, k-means clustering is frequently employed for market segmentation, where customers with similar characteristics or behaviors are grouped together. For instance, a retail company may use k-means clustering to segment its customer base into distinct groups based on factors such as purchasing behavior, demographics, and geographic location. These customer segments can then be targeted with tailored marketing strategies and product offerings to maximize sales and customer satisfaction.Main article: Feature learningk-means clustering has been used as a feature learning (or dictionary learning) step, in either (semi-)supervised learning or unsupervised learning.[54] The basic approach is first to train a k-means clustering representation, using the input training data (which need not be labelled). Then, to project any input datum into the new feature space, an "encoding" function, such as the thresholded matrix-product of the datum with the centroid locations, computes the distance from the datum to each centroid, or simply an indicator function for the nearest centroid,[54][55] or some smooth transformation of the distance.[56] Alternatively, transforming the sample-cluster distance through a Gaussian RBF, obtains the hidden layer of a radial basis function network.[57]This use of k-means has been successfully combined with simple, linear classifiers for semi-supervised learning in NLP (specifically for named-entity recognition)[58] and in computer vision. On an object recognition task, it was found to exhibit comparable performance with more sophisticated feature learning approaches such as autoencoders and restricted Boltzmann machines.[56] However, it generally requires more data, for equivalent performance, because each data point only contributes to one "feature".[54]Example: In natural language processing (NLP), k-means clustering has been integrated with simple linear classifiers for semi-supervised learning tasks such as named-entity recognition (NER). By first clustering unlabeled text data using k-means, meaningful features can be extracted to improve the performance of NER models. For instance, k-means clustering can be applied to identify clusters of words or phrases that frequently co-occur in the input text, which can then be used as features for training the NER model. This approach has been shown to achieve comparable performance with more complex feature learning techniques such as autoencoders and restricted Boltzmann machines, albeit with a greater requirement for labeled data.Recent advancements in the application of k-means clustering include improvements in initialization techniques, such as the use of k-means++ initialization to select initial cluster centroids in a more effective manner. Additionally, researchers have explored the integration of k-means clustering with deep learning methods, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to enhance the performance of various tasks in computer vision, natural language processing, and other domains.Main article: Gaussian mixture modelThe slow "standard algorithm" for k-means clustering, and its associated expectationmaximization algorithm, is a special case of a Gaussian mixture model, specifically, the limiting case when fixing all covariances to be diagonal, equal and have infinitesimal small variance.[59]:850 Instead of small variances, a hard cluster assignment can also be used to show another equivalence of k-means clustering to a special case of "hard" Gaussian mixture modelling.[60]:354,11.4.2.5 This does not mean that it is efficient to use Gaussian mixture modelling to compute k-means, but just that there is a theoretical relationship, and that Gaussian mixture modelling can be interpreted as a generalization of k-means; on the contrary, it has been suggested to use k-means clustering to find starting points for Gaussian mixture modelling on difficult data.[59]:849Main article: k-SVDAnother generalization of the k-means algorithm is the k-SVD algorithm, which estimates data points as a sparse linear combination of "codebook vectors". k-means corresponds to the special case of using a single codebook vector, with a weight of 1.[61]Main article: Principal component analysisThe relaxed solution of k-means clustering, specified by the cluster indicators, is given by principal component analysis (PCA).[62][63] The intuition is that k-means describe spherically shaped (ball-like) clusters. If the data has 2 clusters, the line connecting the two centroids is the best 1-dimensional projection direction, which is also the first PCA direction. Cutting the line at the center of mass separates the clusters (this is the continuous relaxation of the discrete cluster indicator). If the data have three clusters, the 2-dimensional plane spanned by three cluster centroids is the best 2-D projection. This plane is also defined by the first two PCA dimensions. Well-separated clusters are effectively modelled by ball-shaped clusters and thus discovered by k-means. Non-ball-shaped clusters are hard to separate when they are close. For example, two half-moon shaped clusters intertwined in space do not separate well when projected onto PCA subspace. k-means should not be expected to do well on this data.[64] It is straightforward to produce counterexamples to the statement that the cluster centroid subspace is spanned by the principal directions.[65]Main article: Mean shiftBasic mean shift clustering algorithms maintain a set of data points the same size as the input data set. Initially, this set is copied from the input set. All points are then iteratively moved towards the mean of the points surrounding them. By contrast, k-means restricts the set of clusters to k clusters, usually much less than the number of points in the input data set, using the mean of all points in the prior cluster that are closer to that point than any other for the centroid (e.g. within the Voronoi partition of each updating point). A mean shift algorithm that is similar then to k-means, called likelihood mean shift, replaces the set of points undergoing replacement by the mean of all points in the input set that are within a given distance of the changing set.[66] An advantage of mean shift clustering over k-means is the detection of an arbitrary number of clusters in the data set, as there is not a parameter determining the number of clusters. Mean shift can be much slower than k-means, and still requires selection of a bandwidth parameter.Main article: Independent component analysisUnder sparsity assumptions and when input data is pre-processed with the whitening transformation, k-means produces the solution to the linear independent component analysis (ICA) task. This aids in explaining the successful application of k-means to feature learning.[67]Main article: Bilateral filterk-means implicitly assumes that the ordering of the input data set does not matter. The bilateral filter is similar to k-means and mean shift in that it maintains a set of data points that are iteratively replaced by means. However, the bilateral filter restricts the calculation of the (kernel weighted) mean to include only points that are close in the ordering of the input data.[66] This makes it applicable to problems such as image denoising, where the spatial arrangement of pixels in an image is of critical importance.The set of squared error minimizing cluster functions also includes the k-medoids algorithm, an approach which forces the center point of each cluster to be one of the actual points, i.e., it uses medoids in place of centroids.Different implementations of the algorithm exhibit performance differences, with the fastest on a test data set finishing in 10 seconds, the slowest taking 25,988 seconds (~7 hours).[1] The differences can be attributed to implementation quality, language and compiler differences, different termination criteria and precision levels, and the use of indexes for acceleration.The following implementations are available under Free/Open Source Software licenses, with publicly available source code.Accord.NET contains C# implementations for k-means, k-means++ and k-modes.ALGLIB contains parallelized C++ and C# implementations for k-means and k-means++.AOSP contains a Java implementation for k-means.CrimeStat implements two spatial k-means algorithms, one of which allows the user to define the starting locations.ELKI contains k-means (with Lloyd and MacQueen iteration, along with different initializations such as k-means++ initialization) and various more advanced clustering algorithms.Smile contains k-means and various more other algorithms and results visualization (for java, kotlin and scala).Julia contains a k-means implementation in the JuliaStats Clustering package.KNIME contains nodes for k-means and k-medoids.Mahout contains a MapReduce based k-means.mlpack contains a C++ implementation of k-means.Octave contains k-means.OpenCV contains a k-means implementation.Orange includes a component for k-means clustering with automatic selection of k and cluster silhouette scoring.PSPP contains k-means, The QUICK CLUSTER command performs k-means clustering on the dataset.R contains three k-means variations.SciPy and scikit-learn contain multiple k-means implementations.Spark MLlib implements a distributed k-means algorithm.Torch contains an unsup package that provides k-means clustering.Weka contains k-means and x-means.The following implementations are available under proprietary license terms, and may not have publicly available source code.AyasdiMathematicaMATLABOriginProRapidMinerSAP HANASASSPSSStataK-medoidsBFR algorithmCentroidal Voronoi tessellationCluster analysisDBSCANHead/tail breaksk q-flatsk-means++LindeBuzoGray algorithmSelf-organizing map^ a b Kriegel, Hans-Peter; Schubert, Erich; Zimek, Arthur (2016). "The (black) art of runtime evaluation: Are we comparing algorithms or implementations?". Knowledge and Information Systems. 52 (2): 341378. doi:10.1007/s10115-016-1004-2. ISSN0219-1377. S2CID40772241.^ MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. Vol.1. University of California Press. pp.281297. MR0214227. Zbl0214.46201. Retrieved 2009-04-07.^ Steinhaus, Hugo (1957). "Sur la division des corps matriels en parties". Bull. Acad. Polon. Sci. (in French). 4 (12): 801804. MR0090073. Zbl0079.16403.^ Lloyd, Stuart P. (1957). "Least square quantization in PCM". Bell Telephone Laboratories Paper. Published in journal much later: Lloyd, Stuart P. (1982). "Least squares quantization in PCM" (PDF). IEEE Transactions on Information Theory. 28 (2): 129137. CiteSeerX10.1.1.131.1338. doi:10.1109/TIT.1982.1056489. S2CID10833328. Retrieved 2009-04-15.^ Forgy, Edward W. (1965). "Cluster analysis of multivariate data: efficiency versus interpretability of classifications". Biometrics. 21 (3): 768769. JSTOR2528559.^ Pelleg, Dan; Moore, Andrew (1999). "Accelerating exact k -means algorithms with geometric reasoning". Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. San Diego, California, United States: ACM Press. pp.277281. doi:10.1145/312129.312248. ISBN9781581131437. S2CID13907420.^ MacKay, David (2003). "Chapter 20. An Example Inference Task: Clustering" (PDF). Information Theory, Inference and Learning Algorithms. Cambridge University Press. pp.284292. ISBN978-0-521-64298-9. MR2012999.^ Since the square root is a monotone function, this also is the minimum Euclidean distance assignment.^ a b c d e Hartigan, J. A.; Wong, M. A. (1979). "Algorithm AS 136: A k-Means Clustering Algorithm". Journal of the Royal Statistical Society, Series C. 28 (1): 100108. JSTOR2346830.^ a b Hamerly, Greg; Elkan, Charles (2002). "Alternatives to the k-means algorithm that find better clusterings" (PDF). Proceedings of the eleventh international conference on Information and knowledge management (CIKM).^ Celebi, M. E.; Kingravi, H. A.; Vela, P. A. (2013). "A comparative study of efficient initialization methods for the k-means clustering algorithm". Expert Systems with Applications. 40 (1): 200210. arXiv:1209.1960. doi:10.1016/j.eswa.2012.07.021. S2CID6954668.^ Bradley, Paul S.; Fayyad, Usama M. (1998). "Refining Initial Points for k-Means Clustering". Proceedings of the Fifteenth International Conference on Machine Learning.^ Vattani, A. (2011). "k-means requires exponentially many iterations even in the plane" (PDF). Discrete and Computational Geometry. 45 (4): 596616. doi:10.1007/s00454-011-9340-1. S2CID42683406.^ a b Arthur, David; Manthey, B.; Roeglin, H. (2009). "k-means has polynomial smoothed complexity". Proceedings of the 50th Symposium on Foundations of Computer Science (FOCS). arXiv:0904.1113.^ Aloise, D.; Deshpande, A.; Hansen, P.; Popat, P. (2009). "NP-hardness of Euclidean sum-of-squares clustering". Machine Learning. 75 (2): 245249. doi:10.1007/s10994-009-5103-0.^ Dasgupta, S.; Freund, Y. (July 2009). "Random Projection Trees for Vector Quantization". IEEE Transactions on Information Theory. 55 (7): 322942. arXiv:0805.1390. doi:10.1109/TIT.2009.2021326. S2CID666114.^ Mahajan, Meena; Nimbhorkar, Prajakta; Varadarajan, Kasturi (2009). "The Planar k-Means Problem is NP-Hard". WALCOM: Algorithms and Computation. Lecture Notes in Computer Science. Vol.5431. pp.274285. doi:10.1007/978-3-642-00202-1_24. ISBN978-3-642-00201-4.^ Inaba, M.; Katoh, N.; Imai, H. (1994). Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering. Proceedings of 10th ACM Symposium on Computational Geometry. pp.3329. doi:10.1145/177424.178042.^ Manning, Christopher D.; Raghavan, Prabhakar; Schtze, Hinrich (2008). Introduction to information retrieval. Cambridge University Press. ISBN978-0521865715. OCLC190786122.^ a b Arthur, David; Vassilvitskii, Sergei (2006-01-01). "How slow is the k -means method?". Proceedings of the twenty-second annual symposium on Computational geometry. SCG '06. ACM. pp.144153. doi:10.1145/1137856.1137880. ISBN978-1595933409. S2CID3084311.^ Bhowmick, Abhishek (2009). "A theoretical analysis of Lloyd's algorithm for k-means clustering" (PDF). Archived from the original (PDF) on 2015-12-08. See also here.^ Ding, Yufei; Zhao, Yue; Shen, Xipeng; Musuvathi, Madan; Mytkowiczyear, Todd. "Yinyang K-Means: A Drop-In Replacement of the Classic K-Means with Consistent Speedup" (PDF). Proceedings of the Thirty Second International Conference on Machine Learning (ICML).^ a b Phillips, Steven J. (2002). "Acceleration of K-Means and Related Clustering Algorithms". In Mount, David M.; Stein, Clifford (eds.). Acceleration of k-Means and Related Clustering Algorithms. Lecture Notes in Computer Science. Vol.2409. Springer. pp.166177. doi:10.1007/3-540-45643-0_13. ISBN978-3-540-43977-6.^ a b Elkan, Charles (2003). "Using the triangle inequality to accelerate k-means" (PDF). Proceedings of the Twentieth International Conference on Machine Learning (ICML).^ a b Hamerly, Greg (2010). "Making k-means even faster". Proceedings of the 2010 SIAM International Conference on Data Mining. pp.130140. doi:10.1137/1.9781611972801.12. ISBN978-0-89871-703-7.^ a b Hamerly, Greg; Drake, Jonathan (2015). "Accelerating Lloyd's Algorithm for k-Means Clustering". Partitional Clustering Algorithms. pp.4178. doi:10.1007/978-3-319-09259-1_2. ISBN978-3-319-09258-4.^ Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, Jia Heming, K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data, Information Sciences, Volume 622, 2023, Pages 178210, ISSN 0020-0255, 276. doi:10.1007/BF02289263. S2CID 120467216.^ Schubert, Erich (2023-06-22). "Stop using the elbow criterion for k-means and how to choose the number of clusters instead". ACM SIGKDD Explorations Newsletter. 25 (1): 3642. arXiv:2212.12189. doi:10.1145/3606274.3606278. ISSN 1931-0145.^ Peter J. Rousseeuw (1987). "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis". Computational and Applied Mathematics. 20: 5365. doi:10.1016/0377-0427(87)90125-7.^ Robert Tibshirani; Guenther Walther; Trevor Hastie (2001). "Estimating the number of clusters in a data set via the gap statistic". Journal of the Royal Statistical Society, Series B. 63 (2): 411423. doi:10.1111/1467-9868.00293. S2CID 59738652.^ Davies, David L.; Bouldin, Donald W. (1979). "A Cluster Separation Measure". IEEE Transactions on Pattern Analysis and Machine Intelligence. PAMI-1 (2): 224227. doi:10.1109/TPAMI.1979.4766909. S2CID 13254783.^ Caliski, Tadeusz; Harabasz, Jerzy (1974). "A dendrite method for cluster analysis". Communications in Statistics. 3 (1): 127. doi:10.1080/03610927408827101.^ W. M. Rand (1971). "Objective criteria for the evaluation of clustering methods". Journal of the American Statistical Association. 66 (336). American Statistical Association: 846850. doi:10.2307/2284239. JSTOR 2284239.^ Hubert, L., & Arabie, P. (1985).Hubert, L., & Arabie, P. (1985). Comparing partitions. Journal of Classification, 2(1), 193-218. Kanungo, Tapas; Mount, David M.; Netanyahu, Nathan S.; Piatko, Christine D.; Silverman, Ruth; Wu, Angela Y. (2002). "An efficient k-means clustering algorithm: Analysis and implementation" (PDF). IEEE Transactions on Pattern Analysis and Machine Intelligence. 24 (7): 881892. doi:10.1109/TPAMI.2002.1017616. S2CID12003435. Archived from the original (PDF) on 2009-10-07. Retrieved 2009-04-24.^ Drake, Jonathan (2012). "Accelerated k-means with adaptive distance bounds" (PDF). The 5th NIPS Workshop on Optimization for Machine Learning, OPT2012.^ Dhillon, I. S.; Modha, D. M. (2001). "Concept decompositions for large sparse text data using clustering". Machine Learning. 42 (1): 143175. doi:10.1023/a:1007612920971.^ Steinbach, M.; Karypis, G.; Kumar, V. (2000). ""A comparison of document clustering techniques". In". KDD Workshop on Text Mining. 400 (1): 525526.^ Pelleg, D.; & Moore, A. W. (2000, June). "X-means: Extending k-means with Efficient Estimation of the Number of Clusters Archived 2016-09-09 at the Wayback Machine". In ICML, Vol. 1^ Hamerly, Greg; Elkan, Charles (2004). "Learning the k in k-means" (PDF). Advances in Neural Information Processing Systems. 16: 281.^ Amorim, R. C.; Mirkin, B. (2012). "Minkowski Metric, Feature Weighting and Anomalous Cluster Initialisation in k-Means Clustering". Pattern Recognition. 45 (3): 10611075. doi:10.1016/j.patcog.2011.08.012.^ Amorim, R. C.; Hennig, C. (2015). "Recovering the number of clusters in data sets with noise features using feature rescaling factors". Information Sciences. 324: 126145. arXiv:1602.06989. doi:10.1016/j.ins.2015.06.039. S2CID315803.^ Sculley, David (2010). "Web-scale k-means clustering". Proceedings of the 19th international conference on World Wide Web. ACM. pp.11771178. Retrieved 2016-12-21.^ Telgarsky, Matus. "Hartigan's Method: k-means Clustering without Voronoi" (PDF).^ Picciallli, Veronica; Sudoso, Antonio M.; Wiegele, Angelika (2022-03-28). "SOS-SDP: An Exact Solver for Minimum Sum-of-Squares Clustering". INFORMS Journal on Computing. 34 (4): 21442162. arXiv:2104.11542. doi:10.1287/ijoc.2022.1166. ISSN1091-9856. S2CID233388043.^ Bagirov, A. M.; Taheri, S.; Ugon, J. (2016). "Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems". Pattern Recognition. 53: 1224. Bibcode:2016PatRe..53...12B. doi:10.1016/j.patcog.2015.11.011.^ Frnti, Pasi (2018). "Efficiency of random swap clustering". Journal of Big Data. 5 (1): 121. doi:10.1186/s40537-018-0122-y.^ Hansen, P.; Mladenovic, N. (2001). "J-Means: A new local search heuristic for minimum sum of squares clustering". Pattern Recognition. 34 (2): 405413. Bibcode:2001PatRe..34..405H. doi:10.1016/S0031-3203(99)00216-2.^ Krishna, K.; Murty, M. N. (1999). "Genetic k-means algorithm". IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics. 29 (3): 433439. doi:10.1109/3477.764879. PMID18252317.^ a b Gribel, Daniel; Vidal, Thibaut (2019). "HG-means: A scalable hybrid metaheuristic for minimum sum-of-squares clustering". Pattern Recognition. 88: 569583. arXiv:1804.09813. doi:10.1016/j.patcog.2018.12.022. S2CID13746584.^ Mirkes, E. M. "K-means and k-medoids applet". Retrieved 2 January 2016.^ Kulis, Brian; Jordan, Michael I. (2012-06-26). "Revisiting k-means: new algorithms via Bayesian nonparametrics" (PDF). ICML. Association for Computing Machinery. pp.11311138. ISBN9781450312851.^ a b c Coates, Adam; Ng, Andrew Y. (2012). "Learning feature representations with k-means" (PDF). In Montavon, G.; Orr, G. B.; Mller, K.-R. (eds.). Neural Networks: Tricks of the Trade. Springer.^ Csurka, Gabriella; Dance, Christopher C.; Fan, Lixin; Willamowski, Jutta; Bray, Cdric (2004). Visual categorization with bags of keypoints (PDF). ECCV Workshop on Statistical Learning in Computer Vision.^ a b Coates, Adam; Lee, Honglak; Ng, Andrew Y. (2011). An analysis of single-layer networks in unsupervised feature learning (PDF). International Conference on Artificial Intelligence and Statistics (AISTATS). Archived from the original (PDF) on 2013-05-10.^ Schwenker, Friedhelm; Kestler, Hans A.; Palm, Gnther (2001). "Three learning phases for radial-basis-function networks". Neural Networks. 14 (45): 439458. CiteSeerX10.1.1.109.312. doi:10.1016/s0893-6080(01)00027-2. PMID11411631.^ Lin, Dekang; Wu, Xiaoyun (2009). Phrase clustering for discriminative learning (PDF). Annual Meeting of the ACL and IJCNLP. pp.10301038.^ a b Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, B. P. (2007). "Section 16.1. Gaussian Mixture Models and k-Means Clustering". Numerical Recipes: The Art of Scientific Computing (3rded.). New York (NY): Cambridge University Press. ISBN978-0-521-88068-8.^ Kevin P. Murphy (2012). Machine learning: a probabilistic perspective. Cambridge, Mass.: MIT Press. ISBN978-0-262-30524-2. OCLC810414751.^ Aharon, Michal; Elad, Michael; Bruckstein, Alfred (2006). "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation" (PDF). IEEE Transactions on Signal Processing. 54 (11): 4311. Bibcode:2006ITSP...54.4311A. doi:10.1109/TSP.2006.881199. S2CID7477309. Archived from the original (PDF) on 2019-12-02. Retrieved 2019-12-02.^ Zha, Hongyuan; Ding, Chris; Gu, Ming; He, Xiaofeng; Simon, Horst D. (December 2001). "Spectral Relaxation for k-means Clustering" (PDF). Neural Information Processing Systems Vol.14 (NIPS 2001): 10571064.^ Ding, Chris; He, Xiaofeng (July 2004). "K-means Clustering via Principal Component Analysis" (PDF). Proceedings of International Conference on Machine Learning (ICML 2004): 225232.^ Drineas, Petros; Frieze, Alan M.; Kannan, Ravi; Vempala, Santosh; Vinay, Vishwanathan (2004). "Clustering large graphs via the singular value decomposition" (PDF). Machine Learning. 56 (13): 933. doi:10.1023/b:mach.0000033113.59016.96. S2CID5892850. Retrieved 2012-08-02.^ Cohen, Michael B.; Elder, Sam; Musco, Cameron; Musco, Christopher; Persu, Madalina (2014). "Dimensionality reduction for k-means clustering and low rank approximation (Appendix B)". arXiv:1410.6801 [cs.DS].^ a b Little, Max A.; Jones, Nick S. (2011). "Generalized methods and solvers for noise removal from piecewise constant signals. I. Background theory". Proceedings of the Royal Society A. 467 (2135): 30883114. Bibcode:2011RSPSA.467.3088L. doi:10.1098/rspa.2010.0671. PMC3191861. PMID22003312.^ Vinnikov, Alon; Shalev-Shwartz, Shai (2014). "K-means Recovers ICA

Filters when Independent Components are Sparse" (PDF). Proceedings of the International Conference on Machine Learning (ICML 2014).Retrieved from " K-Means Clustering is an Unsupervised Machine Learning algorithm which groups unlabeled dataset into different clusters. It is used to organize data into groups based on their similarity. Understanding K-means ClusteringFor example online store uses K-Means to group customers based on purchase frequency and spending creating segments like Budget Shoppers, Frequent Buyers and Big Spenders for personalised marketing. The algorithm works by first randomly picking some central points called centroids and each data point is then assigned to the closest centroid forming a cluster. After all the points are assigned to a cluster the centroids are updated by finding the average position of the points in each cluster. This process repeats until the centroids stop changing forming clusters. The goal of clustering is to divide the data points into clusters so that similar data points belong to same group. How k-means clustering works?We are given a data set of items with certain features and values for these features like a vector. Thetask is to categorize those items into groups. To achieve this we will use the K-means algorithm.'K' in the name of the algorithm represents the number of groups/clusters we want to classify our items into.K means ClusteringThe algorithm will categorize the items into k groups or clusters of similarity. To calculate that similarity we will use the Euclidean distance as a measurement. The algorithm works as follows:First we randomly initialize k points called means or cluster centroids.We categorize each item to its closest mean and we update the mean's coordinates, which are the averages of the items categorized in that cluster so far.We repeat the process for a given number of iterations and at the end, we have our clusters.The "points" mentioned above are called means because they are the mean values of the items categorized in them. To initialize these means, we have a lot of options. An intuitive method is to initialize the means at random items in the data set. Another method is to initialize the means at random values between the boundaries of the data set. For example for a feature x the items have values in [0,3] we will initialize the means with values for x at [0,3].Selecting the right number of clusters is important for meaningful segmentation to do this we use Elbow Method for optimal value of k in KMeans which is a graphical tool used to determine the optimal number of clusters (k) in K-means.Implementation of K-Means Clustering in PythonWe will use blobs datasets and show how clusters are made.Step 1: Importing the necessary librariesWe are importing Numpy, Matplotlib and scikit learn. Python import numpy as npimport matplotlib.pyplot as pltfrom sklearn.datasets import make_blobsStep 2: Create custom dataset with make_blobs and plot it Python X,y = make_blobs(n_samples = 500,n_features = 2,centers = 3,random_state = 23) fig = plt.figure(0)plt.grid(True)plt.scatter(X[:,0],X[:,1])plt.show()Output:Clustering datasetStep 3: Initializing random centroidsThe code initializes three clusters for K-means clustering. It sets a random seed and generates random cluster centers within a specified range and creates an empty list of points for each cluster. Python k = 3 clusters = {}np.random.seed(23) for idx in range(k): center = 2*(2*np.random.random((X.shape[1],))-1) points = [] cluster = { 'center' : center, 'points' : [] } clusters[idx] = cluster clustersOutput:Random CentroidsStep 4: Plotting random initialize center with data points Python plt.scatter(X[:,0],X[:,1])plt.grid(True)for i in clusters: center = clusters[i]['center'] plt.scatter(center[0],center[1],marker = '*',c = 'red')plt.show()Output:Data points with random centerThe plot displays a scatter plot of data points (X[:,0], X[:,1]) with grid lines. It also marks the initial cluster centers (red stars) generated for K-means clustering.Step 5: Defining Euclidean distance Python def distance(p1,p2): return np.sqrt(np.sum((p1-p2)**2))This step assigns data points to the nearest cluster center and the M-step updates cluster centers based on the mean of assigned points in K-means clustering. Python def assign_clusters(X, clusters): for idx in range(X.shape[0]): dist = [] curr_x = X[idx] for i in range(k): dis = distance(curr_x,clusters[i]['center']) dist.append(dis) curr_cluster = np.argmin(dist) clusters[curr_cluster]['points'].append(curr_x) return clusters def update_clusters(X, clusters): for i in range(k): points = np.array(clusters[i]['points']) if points.shape[0] > 0: new_center = points.mean(axis =0) clusters[i]['center'] = new_center clusters[i]['points'] = [] return clustersStep 7: Creating function to Predict the cluster for the datapoints Python def pred_cluster(X, clusters): pred = [] for i in range(X.shape[0]): dist = [] for j in range(k): dist.append(distance(X[i],clusters[j]['center'])) pred.append(np.argmin(dist)) return pred Python clusters = assign_clusters(X,clusters)clusters = update_clusters(X,clusters)pred = pred_cluster(X,clusters)Step 9: Plotting data points with their predicted cluster center Python plt.scatter(X[:,0],X[:,1],c = pred)for i in clusters: center = clusters[i]['center'] plt.scatter(center[0],center[1],marker = '^',c = 'red')plt.show()Output:K-means ClusteringThe plot shows data points colored by their predicted clusters. The red markers represent the updated cluster centers after the E-M steps in the K-means clustering algorithm. K-Means Clustering Algorithm K-Means Clustering Implementation Machine Learning Tutorial Machine Learning with Python Tutorial Pandas Tutorial NumPy Tutorial - Python Library Scikit Learn Tutorial ML | Data Preprocessing in Python EDA - Exploratory Data Analysis in Python Every machine Learning engineer wants to achieve accurate predictions with their algorithms. Such learning algorithms are generally divided into two types: supervised and unsupervised. K-means clustering is one of the unsupervised algorithms where the available input data does not have a labeled response.What is ClusteringClustering is like sorting a bunch of similar items into different groups based on their characteristics. In data mining and machine learning, its a powerful technique used to group similar data points together, making it easier to find patterns or understand large datasets. Essentially, clustering helps identify natural groupings in your data. There are two common types of clustering methods:Types of ClusteringClustering is a type of unsupervised learning wherein data points are grouped into different sets based on their degree of similarity.The various types of clustering are:Hierarchical clusteringPartitioning clusteringHierarchical clustering is further subdivided into:Agglomerative clusteringDivisive clusteringPartitioning clustering is further subdivided into:K-Means clusteringFuzzy C-Means clusteringHierarchical ClusteringHierarchical clustering uses a tree-like structure, like so:In agglomerative clustering, there is a bottom-up approach. We begin with each element as a separate cluster and merge them into successively more massive clusters, as shown below:Divisive clustering is a top-down approach. We begin with the whole set and proceed to divide it into successively smaller clusters, as you can see below:Partitioning ClusteringPartitioning clustering is split into two subtypes - K-Means clustering and Fuzzy C-Means.In k-means clustering, the objects are divided into several clusters mentioned by the number K. So if we say K = 2, the objects are divided into two clusters, c1 and c2, as shown:Here, the features or characteristics are compared, and all objects having similar characteristics are clustered together.Fuzzy c-means is very similar to k-means in the sense that it clusters objects that have similar characteristics together. In k-means clustering, a single object cannot belong to two different clusters. But in c-means, objects can belong to more than one cluster, as shown.What is K-Means Clustering?K-means clustering is a way of grouping data based on how similar or close the data points are to each other. Imagine you have a bunch of points, and you want to group them into clusters. The algorithm works by first randomly picking some central points (called centroids) and then assigning every data point to the nearest centroid.Once thats done, it recalculates the centroids based on the new groupings and repeats the process until the clusters make sense. Its a pretty fast and efficient method, but it works best when the clusters are distinct and not too mixed up. One challenge, though, is figuring out the right number of clusters (K) beforehand. Plus, if theres a lot of noise or overlap in the data, K Means might not perform as well.Objective of K-Means ClusteringK-Means clustering primarily aims to organize similar data points into distinct groups. Heres a look at its key objectives:K-Means is designed to cluster data points that share common traits, allowing patterns or trends to emerge. Whether analyzing customer behavior or images, the method helps reveal hidden relationships within your dataset.Another objective is to keep data points in each group as close to the cluster's centroid as possible. Reducing this internal distance results in compact, cohesive clusters, enhancing the accuracy of your results.K-Means also aims to maintain clear separation between different clusters. By maximizing the distance between groups, the algorithm ensures that each cluster remains distinct, providing a better understanding of data categories without overlap.Properties of K-Means ClusteringNow, lets look at the key properties that make K-means clustering algorithm effective in creating meaningful groups:One of the main things K Means aims for is that all the data points in a cluster should be pretty similar to each other. Imagine a bank that wants to group its customers based on income and debt. If customers within the same cluster have vastly different financial situations, then a one-size-fits-all approach to offers might not work. For example, a customer with high income and high debt might have different needs compared to someone with low income and low debt. By making sure the customers in each cluster are similar, the bank can create more tailored and effective strategies.Another important aspect is that the clusters themselves should be as distinct from each other as possible. Going back to our bank example, if one cluster consists of high-income, high-debt customers and another cluster has high-income, low-debt customers, the differences between the clusters are clear. This separation helps the bank create different strategies for each group. If the clusters are too similar, it can be challenging to treat them as separate segments, which can make targeted marketing less effective.Applications of K-Means ClusteringHere are some interesting ways K-means clustering is put to work across different fields:At the heart of K-Means clustering is the concept of distance. Euclidean distance, for example, is a simple straight-line measurement between points and is commonly used in many applications. Manhattan distance, however, follows a grid-like path, much like how you'd navigate city streets. Squared Euclidean distance makes calculations easier by squaring the values, while cosine distance is handy when working with text data because it measures the angle between data vectors. Picking the right distance measure really depends on what kind of problem youre solving and the nature of your data.K-Means clustering has even been applied to studying the eruptions of the Old Faithful geyser in Yellowstone. The data collected includes eruption duration and the waiting time between eruptions. By clustering this information, researchers can uncover patterns that help predict the geysers behavior. For instance, you might find clusters of similar eruption durations and intervals, which could improve predictions for future eruptions.One of the most popular uses of K-means clustering is for customer segmentation. From banks to e-commerce, businesses use K-means clustering customer segmentation to group customers based on their behaviors. For example, in telecom or sports industries, companies can create targeted marketing campaigns by understanding different customer segments better. This allows for personalized offers and communications, boosting customer engagement and satisfaction.When dealing with a vast collection of documents, K-Means can be a lifesaver. It groups similar documents together based on their content, which makes it easier to manage and retrieve relevant information. For instance, if you have thousands of research papers, clustering can quickly help you find related studies, improving both organization and efficiency in accessing valuable information.In image processing, K-Means clustering is commonly used to group pixels with similar colors, which divides the image into distinct regions. This is incredibly helpful for tasks like object detection and image enhancement. For instance, clustering can help separate objects within an image, making analysis and processing more accurate. Its also widely used to extract meaningful features from images in various visual tasks.K-Means clustering also plays a vital role in recommendation systems. Say you want to suggest new songs to a listener based on their past preferences; clustering can group similar songs together, helping the system provide personalized suggestions. By clustering content that shares similar features, recommendation engines can deliver a more tailored experience, helping users discover new songs that match their taste.K-Means can even help with image compression by reducing the number of colors in an image while keeping the visual quality intact. K-Means reduces the image size without losing much detail by clustering similar colors and replacing the pixels with the average of their cluster. Its a practical method for compressing images for more accessible storage and transmission, all while maintaining visual clarity.Advantages of K-meansSimple and easy to implement: The k-means algorithm is easy to understand and implement, making it a popular choice for clustering tasks.Fast and efficient: K-means is computationally efficient and can handle large datasets with high dimensionality.Scalability: K-means can handle large datasets with many data points and can be easily scaled to handle even larger datasets.Flexibility: K-means can be easily adapted to different applications and can be used with varying metrics of distance and initialization methods.Disadvantages of K-MeansSensitivity to initial centroids: K-means is sensitive to the initial selection of centroids and can converge to a suboptimal solution.Requires specifying the number of clusters: The number of clusters k needs to be specified before running the algorithm, which can be challenging in some applications.Sensitive to outliers: K-means is sensitive to outliers, which can have a significant impact on the resulting clusters.When it comes to evaluating how well your clustering algorithm is working, there are a few key metrics that can help you get a clearer picture of your results. Heres a rundown of the most useful ones:Silhouette analysis is like a report card for your clusters. It measures how well each data point fits into its own cluster compared to other clusters. A high silhouette score means that your points are snugly fitting into their clusters and are quite distinct from points in other clusters. Imagine a score close to 1 as a sign that your clusters are well-defined and separated. Conversely, a score close to 0 indicates some overlap, and a negative score suggests that the clustering might need some work.Inertia is a bit like a gauge of how tightly packed your data points are within each cluster. It calculates the sum of squared distances from each point to the cluster's center (or centroid). Think of it as measuring how snugly the points are huddled together. Lower inertia means that points are closer to the centroid and to each other, which generally indicates that your clusters are well-formed. For most numeric data, you'll use Euclidean distance, but if your data includes categorical features, Manhattan distance might be better.The Dunn Index takes a broader view by considering both the distance within and between clusters. Its calculated as the ratio of the smallest distance between any two clusters (inter-cluster distance) to the largest distance within a cluster (intra-cluster distance). A higher Dunn Index means that clusters are not only tight and cohesive internally but also well-separated from each other. In other words, you want your clusters to be as far apart as possible while being as compact as possible.How Does K-Means Clustering Work?The flowchart below shows how k-means clustering works:The goal of the K-Means algorithm is to find clusters in the given input data. There are a couple of ways to accomplish this. We can use the trial and error method by specifying the value of K (e.g., 3,4, 5). As we progress, we keep changing the value until we get the best clusters.Another method is to use the Elbow technique to determine the value of K. Once we get the K's value, the system will assign that many centroids randomly and measure the distance of each of the data points from these centroids. Accordingly, it assigns those points to the corresponding centroid from which the distance is minimum. So each data point will be assigned to the centroid, which is closest to it. Thereby we have a K number of initial clusters.It calculates the new centroid position for the newly formed clusters. The centroid's position moves compared to the randomly allocated one.Once again, the distance of each point is measured from this new centroid point. If required, the data points are relocated to the new centroids, and the mean position or the new centroid is calculated once again.If the centroid moves, the iteration continues indicating no convergence. But once the centroid stops moving (which means that the clustering process has converged), it will reflect the result.Let's use a visualization example to understand this better.We have a data set for a grocery shop, and we want to find out how many clusters this has to be spread across. To find the optimum number of clusters, we break it down into the following steps:Step 1:The Elbow method is the best way to find the number of clusters. The elbow method constitutes running K-Means clustering on the dataset.Next, we use within-sum-of-squares as a measure to find the optimum number of clusters that can be formed for a given data set. Within the sum of squares (WSS) is defined as the sum of the squared distance between each member of the cluster and its centroid.The WSS is measured for each value of K. The value of K, which has the least amount of WSS, is taken as the optimum value.Now, we draw a curve between WSS and the number of clusters.Here, WSS is on the y-axis and number of clusters on the x-axis.You can see that there is a very gradual change in the value of WSS as the K value increases from 2.So, you can take the elbow point value as the optimal value of K. It should be either two, three, or at most four. But, beyond that, increasing the number of clusters does not dramatically change the value in WSS, it gets stabilized.Step 2:Let's assume that these are our delivery points:We can randomly initialize two points called the cluster centroids.Here, C1 and C2 are the centroids assigned randomly.Step 3:Now the distance of each location from the centroid is measured, and each data point is assigned to the centroid, which is closest to it.This is how the initial grouping is done:Step 4:Compute the actual centroid of data points for the first group.Step 5:Reposition the random centroid to the actual centroid.Step 6:Compute the actual centroid of data points for the second group.Step 7:Reposition the random centroid to the actual centroid.Step 8:Once the cluster becomes static, the k-means algorithm is said to be converged.The final cluster with centroids c1 and c2 is as shown below:K-Means Clustering AlgorithmLet's say we have x1, x2, x3 x(n) as our inputs, and we want to split this into K clusters.The steps to form clusters are:Step 1: Choose K random points as cluster centers called centroids.Step 2: Assign each x(i) to the closest cluster by implementing euclidean distance (i.e., calculating its distance to each centroid)Step 3: Identify new centroids by taking the average of the assigned points.Step 4: Keep repeating step 2 and step 3 until convergence is achievedLet's take a detailed look at it at each of these steps.Step 1:We randomly pick K (centroids). We name them c1,c2,..... ck, and we can say thatWhere C is the set of all centroids.Step 2:We assign each data point to its nearest center, which is accomplished by calculating the euclidean distance.Where dist() is the Euclidean distance.Here, we calculate each x value's distance from each c value, i.e. the distance between x1-c1, x1-c2, x1-c3, and so on. Then we find which is the lowest value and assign x1 to that particular centroid.Similarly, we find the minimum distance for x2, x3, etc.Step 3:We identify the actual centroid by taking the average of all the points assigned to that cluster.Where Si is the set of all points assigned to the ith cluster.It means the original point, which we thought was the centroid, will shift to the new position, which is the actual centroid for each of these groups.Step 4:Keep repeating step 2 and step 3 until convergence is achieved.How to Choose the Value of "K number of clusters" in K-Means Clustering?Although many choices are available for choosing the optimal number of clusters, the Elbow Method is one of the most popular and appropriate methods. The Elbow Method uses the idea of WCSS value, which is short for for Within Cluster Sum of Squares. WCSS defines the total number of variations within a cluster. This is the formula used to calculate the value of WCSS (for three clusters) provided courtesy of Javatpoint:WCSS= Pi in Cluster1 distance(Pi C1)2 +Pi in Cluster2distance(Pi C2)2+Pi in CLuster3 distance(Pi C3)2Python Implementation of the K-Means Clustering AlgorithmHeres how to use Python to implement the K-Means Clustering Algorithm. These are the steps you need to take:Data pre-processingFinding the optimal number of clusters using the elbow methodTraining the K-Means algorithm on the training data setVisualizing the clusters1. Data Pre-Processing. Import the libraries, datasets, and extract the independent variables.# importing librariesimport numpy as nmimport matplotlib.pyplot as mtpimport pandas as pd# Importing the datasetdataset = pd.read_csv('Mall_Customers_data.csv')x = dataset.iloc[:, [3, 4]].values2. Find the optimal number of clusters using the elbow method. Heres the code you use:#finding optimal number of clusters using the elbow methodfrom sklearn.cluster import KMeanswcss_list= [] #Initializing the list for the values of WCSS#Using for loop for iterations from 1 to 10.for i in range(1, 11):kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)kmeans.fit(x)wcss_list.append(kmeans.inertia_)mtp.plot(range(1, 11), wcss_list)mtp.title('The Elobw Method Graph')mtp.xlabel('Number of clusters(k)') mtp.ylabel('wcss_list')mtp.show()3. Train the K-means algorithm on the training dataset. Use the same two lines of code used in the previous section. However, instead of using i, use 5, because there are 5 clusters that need to be formed. Heres the code:#training the K-means model on a datasetkmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)y_predict= kmeans.fit_predict(x)4. Visualize the Clusters. Since this model has five clusters, we need to visualize each one.#visulaizing the clustersmtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1') #for first clustermtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2') #for second clustermtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3') #for third clustermtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') #for fourth clustermtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5') #for fifth clustermtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroid')mtp.title('Clusters of customers')mtp.xlabel('Annual Income (k$)')mtp.ylabel('Spending Score (1-100)')mtp.legend()mtp.show()Challenges With K-Means Clustering AlgorithmK-Means is a powerful tool, but its not without its hiccups. Here are a couple of common challenges you might face:One issue you might run into with K Means is when clusters vary in size. Picture this: you have clusters that are small and spread out on the edges, and a larger, more central cluster. When K Means is applied, it can struggle to evenly distribute the data. The algorithm might create clusters that dont quite match the actual data distribution, leading to some clusters being too small or too large compared to others.Another challenge comes up when the clusters have different densities. Imagine you have some clusters with tightly packed points and others where the points are more scattered. K Means might have trouble with this. It tends to group points based on distance from the cluster center, so tightly packed points might end up in one cluster, while scattered points could be split across different clusters, even if they actually belong together. This can lead to clusters that dont accurately reflect the true structure of your data.Demo: K-Means ClusteringProblem StatementWalmart wants to open a chain of stores across the state of Florida and find the optimal store locations to maximize revenue.The issue here is that if they open too many stores close to each other, they will not make a profit. But if the stores are too far apart, they will not have enough sales coverage.SolutionWalmart is an e-commerce giant. Its database already contains customers' addresses, which it can use to perform K-Means Clustering to find the optimal location. K-means, proposed by Stuart Lloyd in 1957, is one of the most widely used unsupervised learning algorithms. It iteratively partitions data into K non-overlapping clusters, maximizing intra-cluster similarity and inter-cluster differences. Here, K denotes the predefined number of clusters, and "means" refers to the centroid (mean) of data points within each cluster. The phrase "birds of a feather flock together" aptly describes clustering. However, it is crucial to distinguish clustering from classification: Classification involves predefined labels (supervised learning), where data is assigned to known categories. Clustering identifies hidden patterns in unlabeled data (unsupervised learning), grouping similar data points without prior knowledge of categories. Applications of K-means Clustering K-means is versatile across various domains: Gene Expression Analysis: Clustering co-expressed genes to identify functional modules (e.g., cancer-related gene clusters). For example, RNA-seq data can group genes with similar expression patterns for GO/KEGG enrichment analysis. Single-Cell Sequencing: Classifying cell subtypes (e.g., T cells, B cells) and annotating them. Tools like t-SNE or UMAP are often integrated for dimensionality reduction and visualizing cellular heterogeneity. Proteomics: Clustering protein sequences or structures to predict conserved functional regions. Case studies include classifying proteins based on physicochemical properties (hydrophobicity, charge) to aid functional annotation. Disease Subtyping: Stratifying patients into subtypes using multi-omics data (genomics, transcriptomics) to guide personalized therapies. The K-means Algorithm: Step-by-Step The algorithm consists of two core steps: Assignment and Update. 1. Initialize Centroids: Randomly select K data points as initial centroids. 2. Assignment Step: Assign each data point to the nearest centroid using Euclidean distance. 3. Update Step: Recalculate centroids as the mean of all points in each cluster. 4. Iterate: Repeat assignment and update until centroids stabilize (convergence) or a maximum iteration threshold is reached. Visualizing Iterations: Initially, centroids (black dots) are randomly placed. As iterations proceed, centroids shift, and cluster assignments (color-coded) adjust until convergence. Iterative process for K-means analysis Example: Clustering Data Consider 2D data points representing customer "spending" and "visit frequency." We aim to cluster them into *K=2* groups. Data Point Spending (x) Visits (y) Point 1 1 2 Point 2 2 1 Point 3 4 5 Point 4 5 4 Point 5 8 8 Step 1: Initialize Centroids Randomly select Point 1 (1,2) and Point 5 (8,8) as initial centroids: Step 2: Assign Clusters Calculate Euclidean distances and assign points to the nearest centroid: Point Distance to C1 Distance to C2 Centroids Cluster 1 0 8.49 C1 C1 2 1.41 8.06 C1 C1 3 4.24 5 C1 C1 4 5 4.24 C2 C2 5 8.49 0 C2 C2 Result: Cluster 1: Points 1, 2, 3 Cluster 2: Points 4, 5 Step 3: Recalculate Centroids New C1: ((1+2+4)/3, (2+1+5)/3) = (2.33, 2.67) New C2: ((5+8)/2, (4+8)/2) = (6.5, 6.0) Step 4: Iterate Until Convergence Repeat assignment and centroid updates until no further changes occur. Final clusters remain unchanged, with stable centroids at (2.33, 2.67) and (6.5, 6.0). Choosing the Optimal K Value Selecting K is critical but non-trivial. Common methods include: 1. Elbow Method: Plot the sum of squared errors (SSE) against K. The "elbow" (sharp decline in SSE slope) indicates the optimal K. 2. Silhouette Coefficient: Measures cluster compactness and separation. Higher values denote better clustering. 3. Calinski-Harabasz Index: Ratio of between-cluster to within-cluster variance. Maximize this value. K-means Analysis & Visualization 1.Environment Setup #Install required R packages: if (!requireNamespace("factoextra", quietly = TRUE)) { install.packages("factoextra") } if (!requireNamespace("cluster", quietly = TRUE)) { install.packages("cluster") } 2.Data Preparation Use the `USArrests` dataset (crime statistics by U.S. state): library(factoextra) library(cluster) data