How can I perform K-Means clustering in Python using a step-by-step example?

Authored by

stats writer

June 27, 2024

RECOMMENDED CITATION

stats writer (2024). *How can I perform K-Means clustering in Python using a step-by-step example?*. PSYCHOLOGICAL SCALES. Retrieved from https://scales.arabpsychology.com/?p=154515

ARABPSYCHOLOGY.COM

K-Means clustering is a popular unsupervised machine learning algorithm used to group data points into a specified number of clusters. In Python, this algorithm can be implemented using the scikit-learn library.

To perform K-Means clustering in Python, follow these steps:

1. Import the necessary libraries, including pandas for data manipulation, matplotlib for data visualization, and sklearn for the K-Means algorithm.
2. Load the data set into a pandas DataFrame and preprocess it if needed.
3. Choose the number of clusters to be formed and initialize the K-Means algorithm by creating an instance of the KMeans class.
4. Fit the data to the algorithm by passing the DataFrame and the number of clusters as parameters.
5. Use the fitted model to predict the cluster labels for each data point.
6. Visualize the results by plotting the data points with their respective cluster labels.
7. Evaluate the performance of the clustering by calculating metrics such as the within-cluster sum of squares and silhouette score.
8. Repeat the process with different values for the number of clusters until the optimal number is determined.

By following these step-by-step instructions and utilizing the scikit-learn library, K-Means clustering can be easily performed in Python to group data points into distinct clusters.

K-Means Clustering in Python: Step-by-Step Example

One of the most common clustering algorithms in is known as k-means clustering.

K-means clustering is a technique in which we place each observation in a dataset into one of *K* clusters.

The end goal is to have *K* clusters in which the observations within each cluster are quite similar to each other while the observations in different clusters

ARABPSYCHOLOGY.COM

are quite different from each other.

In practice, we use the following steps to perform K-means clustering:

1. Choose a value for *K*.

First, we must decide how many clusters we'd like to identify in the data. Often we have to simply test several different values for *K* and analyze the results to see which number of clusters seems to make the most sense for a given problem.

2. Randomly assign each observation to an initial cluster, from 1 to *K*.

3. Perform the following procedure until the cluster assignments stop changing.

For each of the *K* clusters, compute the cluster *centroid.* This is simply the vector of the *p* feature means for the observations in the *k*th cluster.Assign each observation to the cluster whose centroid is closest. Here, *closest* is defined using Euclidean distance.

ARABPSYCHOLOGY.COM

The following step-by-step example shows how to perform k-means clustering in Python by using the KMeans function from the sklearn module.

Step 1: Import Necessary Modules

First, we'll import all of the modules that we will need to perform k-means clustering:

```
import pandas as pd
import numpy as np
import matplotlib.pyplotas plt
from sklearn.clusterimport KMeans
from sklearn.preprocessingimport StandardScaler
```

Step 2: Create the DataFrame

Next, we'll create a DataFrame that contains the following three variables for 20 different basketball players:

pointsassistsrebounds

The following code shows how to create this pandas DataFrame:

```
#create DataFrame
```

ARABPSYCHOLOGY.COM

```
df = pd.DataFrame({'points': ,
'assists': ,
'rebounds': })

#view first five rows of DataFrame
print(df.head())

points assists rebounds
0 18.0 3.0 15
1 NaN 3.0 14
2 19.0 4.0 14
3 14.0 5.0 10
4 14.0 4.0 8
```

We will use k-means clustering to group together players that are similar based on these three metrics.

Step 3: Clean & Prep the DataFrame

Next, we'll perform the following steps:

Use dropna() to drop rows with NaN values in any columnUse StandardScaler() to scale each variable to have a mean of 0 and a standard deviation of 1

The following code shows how to do so:

ARABPSYCHOLOGY.COM

#drop rows with NA values in any columns
df = df.dropna()

#create scaled DataFrame where each variable has mean of 0 and standard dev of 1
scaled_df = StandardScaler().fit_transform(df)

#view first five rows of scaled DataFrameprint(scaled_df)

]

Note: We use scaling so that each variable has equal importance when fitting the k-means algorithm. Otherwise, the variables with the widest ranges would have too much influence.

Step 4: Find the Optimal Number of Clusters

To perform k-means clustering in Python, we can use the KMeans function from the sklearn module.

This function uses the following basic syntax:

KMeans(init='random', n_clusters=8, n_init=10, random_state=None)

ARABPSYCHOLOGY.COM

where:

init: Controls the initialization technique.n_clusters: The number of clusters to place observations in.n_init: The number of initializations to perform. The default is to run the k-means algorithm 10 times and return the one with the lowest SSE.random_state: An integer value you can pick to make the results of the algorithm reproducible.

The most important argument in this function is n_clusters, which specifies how many clusters to place the observations in.

However, we don't know beforehand how many clusters is optimal so we must create a plot that displays the number of clusters along with the SSE (sum of squared errors) of the model.

Typically when we create this type of plot we look for an "elbow" where the sum of squares begins to "bend" or level off. This is typically the optimal number of clusters.

The following code shows how to create this type of

ARABPSYCHOLOGY.COM

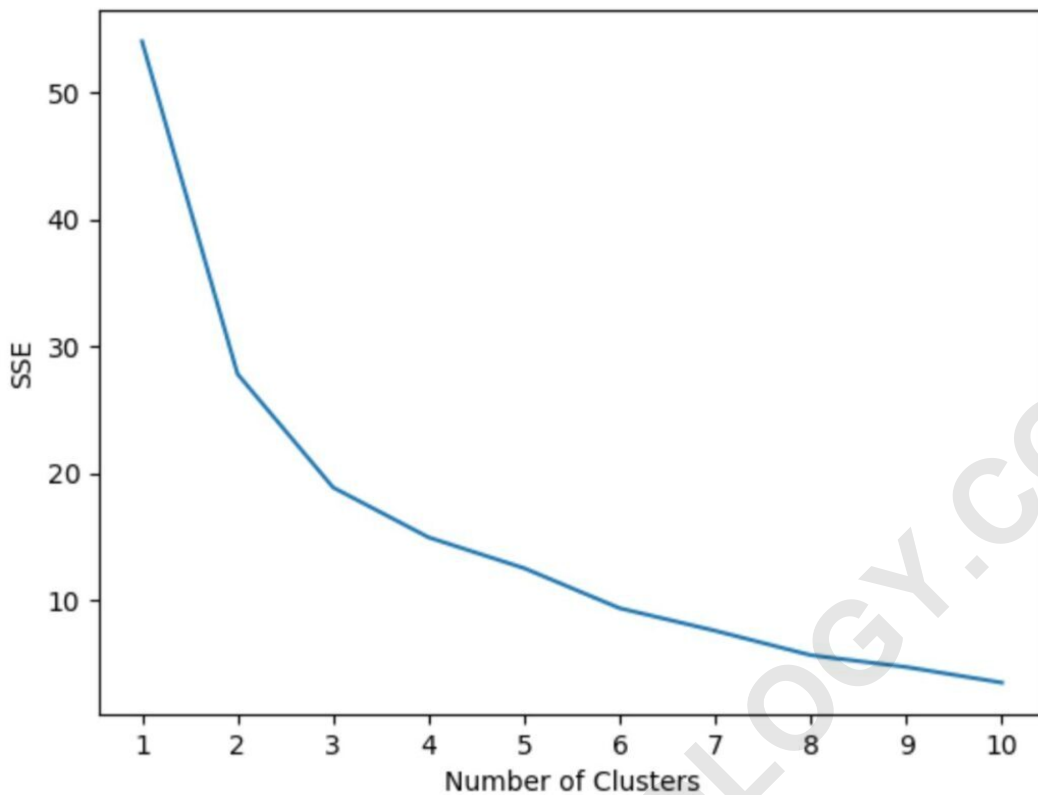
plot that displays the number of clusters on the x-axis and the SSE on the y-axis:

```
#initialize kmeans parameters
kmeans_kwargs = {
"init": "random",
"n_init": 10,
"random_state": 1,
}

#create list to hold SSE values for each k
sse =
for k in range(1, 11):
kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
kmeans.fit(scaled_df)
sse.append(kmeans.inertia_)

#visualize results
plt.plot(range(1, 11), sse)
plt.xticks(range(1, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.show()
```

ARABPSYCHOLOGY.COM



ARABPSYCHOLOGY.COM

In this plot it appears that there is an elbow or "bend" at k = 3 clusters.

Thus, we will use 3 clusters when fitting our k-means clustering model in the next step.

Note: In the real-world, it's recommended to use a combination of this plot along with domain expertise to pick how many clusters to use.

Step 5: Perform K-Means Clustering with Optimal *K*

The following code shows how to perform k-means

clustering on the dataset using the optimal value for *k* of 3:

```
#instantiate the k-means class, using optimal number of clusters
kmeans = KMeans(init="random", n_clusters=3, n_init=10, random_state=1)

#fit k-means algorithm to data
kmeans.fit(scaled_df)

#view cluster assignments for each observation
kmeans.labels_

array()
```

The resulting array shows the cluster assignments for each observation in the DataFrame.

To make these results easier to interpret, we can add a column to the DataFrame that shows the cluster assignment of each player:

```
#append cluster assingments to original DataFrame
df = kmeans.labels_#view updated DataFrame
```

ARABPSYCHOLOGY.COM

print(df)

points assists rebounds cluster
0 18.0 3.0 15 1
2 19.0 4.0 14 1
3 14.0 5.0 10 1
4 14.0 4.0 8 1
5 11.0 7.0 14 1
6 20.0 8.0 13 1
7 28.0 7.0 9 2
8 30.0 6.0 5 2
9 31.0 9.0 4 0
10 35.0 12.0 11 0
11 33.0 14.0 6 0
13 25.0 9.0 5 0
14 25.0 4.0 3 2
15 27.0 3.0 8 2
16 29.0 4.0 12 2
17 30.0 12.0 7 0
18 19.0 15.0 6 0
19 23.0 11.0 5 0

The cluster column contains a cluster number (0, 1, or 2) that each player was assigned to.

ARABPSYCHOLOGY.COM

Players that belong to the same cluster have roughly similar values for the points, assists, and rebounds columns.

Note: You can find the complete documentation for the KMeans function from sklearn .

The following tutorials explain how to perform other common tasks in Python:

ARABPSYCHOLOGY.COM