KModes Clustering Algorithm for Categorical data

Harika Bonthu — June 13, 2021

Algorithm Beginner Clustering Data Science Python Unsupervised

This article was published as a part of the Data Science Blogathon

Introduction:

Clustering is an unsupervised learning method whose task is to divide the population or data points into a number of groups, such that data points in a group are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects based on similarity and dissimilarity between them.

KModes clustering is one of the unsupervised Machine Learning algorithms that is used to cluster **categorical variables.**

You might be wondering, why KModes when we already have KMeans.

KMeans uses mathematical measures (distance) to cluster continuous data. The lesser the distance, the more similar our data points are. Centroids are updated by Means.
But for categorical data points, we cannot calculate the distance. So we go for KModes algorithm. It uses the dissimilarities(total mismatches) between the data points. The lesser the dissimilarities the more similar our data points are. It uses Modes instead of means.

In this blog, we will learn:

- How does the KModes algorithm work?
- Implementation of KModes in Python

1. Pick K observations at random and use them as leaders/clusters
2. Calculate the dissimilarities and assign each observation to its closest cluster
3. Define new modes for the clusters
4. Repeat 2–3 steps until there are is no re-assignment required

I hope you got the basic idea of the KModes algorithm by now. So let us quickly take an example to illustrate the working step by step.

Example: Imagine we have a dataset that has the information about hair color, eye color, and skin color of persons. We aim to group them based on the available information(maybe we want to suggest some styling ideas)

Hair color, eye color, and skin color are all categorical variables. Below 👇 is how our dataset looks like.

person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Image of our data

Alright, we have the sample data now. Let us proceed by defining the number of clusters(K)=3

Step 1: Pick K observations at random and use them as leaders/clusters

I am choosing P1, P7, P8 as leaders/clusters

Leaders			
P1	blonde	amber	fair
P7	red	green	fair
P8	black	hazel	fair

person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Leaders and Observations

Step 2: Calculate the dissimilarities(no. of mismatches) and assign each observation to its closest cluster

Iteratively compare the cluster data points to each of the observations. Similar data points give 0, dissimilar data points give 1.

Leaders			
P1	blonde	amber	fair
P7	red	green	fair
P8	black	hazel	fair

person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Comparing leader/Cluster P1 to the observation P1 gives 0 dissimilarities.

Leaders			
P1	blonde	amber	fair
P7	red	green	fair
P8	black	hazel	fair

person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Comparing leader/cluster P1 to the observation P2 gives 3(1+1+1) dissimilarities.

Likewise, calculate all the dissimilarities and put them in a matrix as shown below and assign the observations to their closest cluster(cluster that has the least dissimilarity)

	Cluster 1 (P1)	Cluster 2 (P7)	Cluster 3 (P8)	Cluster
P1	0 ✓	2	2	Cluster 1
P2	3 ✓	3	3	Cluster 1
P3	3	1 ✓	3	Cluster 2
P4	3	3	1 ✓	Cluster 3
P5	1 ✓	2	2	Cluster 1
P6	3	3	2 ✓	Cluster 3
P7	2	0 ✓	2	Cluster 2
P8	2	2	0 ✓	Cluster 3

Dissimilarity matrix (Image by Author)

After step 2, the observations P1, P2, P5 are assigned to cluster 1; P3, P7 are assigned to Cluster 2; and P4, P6, P8 are assigned to cluster 3.

Step 3: Define new modes for the clusters

Mode is simply the **most observed value**.

Mark the observations according to the cluster they belong to. Observations of Cluster 1 are marked in Yellow, Cluster 2 are marked in Brick red, and Cluster 3 are marked in Purple.

person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Looking for Modes (Image by author)

Considering one cluster at a time, for each feature, look for the Mode and update the new leaders.

Explanation: Cluster 1 observations(P1, P2, P5) has brunette as the most observed hair color, amber as the most observed eye color, and fair as the most observed skin color.

Note: If you observe the same occurrence of values, take the mode randomly. In our case, the observations of Cluster 3(P3, P7) have one occurrence of brown, fair skin color. I randomly chose brown as the mode.

Below are our new leaders after the update.

New Leaders			
	hair color	eye color	skin color
Cluster 1	brunette	amber	fair
Cluster 2	red	green	fair
Cluster 3	black	hazel	brown

Obtained new leaders

Repeat steps 2–4

After obtaining the new leaders, again calculate the dissimilarities between the observations and the newly obtained leaders.

New Leaders			
	hair color	eye color	skin color
Cluster 1	brunette	amber	fair
Cluster 2	red	green	fair
Cluster 3	black	hazel	brown

person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown

Comparing Cluster 1 to the observation P1 gives 1 dissimilarity.

New Leaders			
	hair color	**eye color**	**skin color**
Cluster 1	brunette	amber	fair
Cluster 2	red	green	fair
Cluster 3	black	hazel	brown
person	**hair color**	**eye color**	**skin color**
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Comparing Cluster 1 to the observation P2 gives 2 dissimilarities.

Likewise, calculate all the dissimilarities and put them in a matrix. Assign each observation to its closest cluster.

	Cluster 1	Cluster 2	Cluster 3	Cluster
P1	1 ✓	2	3	Cluster 1
P2	2 ✓	3	2	Cluster 1
P3	3	1 ✓	2	Cluster 2
P4	3	3	0 ✓	Cluster 3
P5	0 ✓	2	3	Cluster 1
P6	3	3	1 ✓	Cluster 3
P7	2	0 ✓	3	Cluster 2
P8	2	2	1 ✓	Cluster 3

The observations P1, P2, P5 are assigned to Cluster 1; P3, P7 are assigned to Cluster 2; and P4, P6, P8 are assigned to Cluster 3.

We stop here as we see there is no change in the assignment of observations.

Implementation of KModes in Python:

Begin with Importing necessary libraries

```
# importing necessary libraries
import pandas as pd
import numpy as np
# !pip install kmodes
from kmodes.kmodes import KModes
import matplotlib.pyplot as plt
%matplotlib inline
```

Creating toy dataset

```
# Create toy dataset
hair_color = np.array(['blonde', 'brunette', 'red', 'black', 'brunette', 'black', 'red', 'black'])
eye_color = np.array(['amber', 'gray', 'green', 'hazel', 'amber', 'gray', 'green', 'hazel'])
skin_color = np.array(['fair', 'brown', 'brown', 'brown', 'fair', 'brown', 'fair', 'fair'])
person = ['P1','P2','P3','P4','P5','P6','P7','P8']
data = pd.DataFrame({'person':person, 'hair_color':hair_color, 'eye_color':eye_color,
'skin_color':skin_color})
data = data.set_index('person')
data
```

person	hair_color	eye_color	skin_color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Image of our dataset

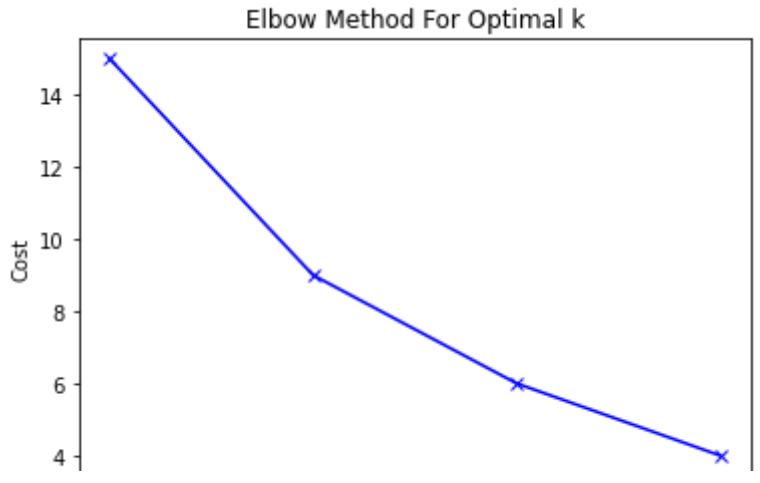
Scree Plot or Elbow curve to find optimal K value

For KModes, plot cost for a range of K values. Cost is the sum of all the dissimilarities between the clusters.

Select the K where you observe an elbow-like bend with a lesser cost value.

```
# Elbow curve to find optimal K
cost = []
K = range(1,5)
for num_clusters in list(K):
    kmode = KModes(n_clusters=num_clusters, init = "random", n_init = 5, verbose=1)
    kmode.fit_predict(data)
    cost.append(kmode.cost_)


plt.plot(K, cost, 'bx-')
plt.xlabel('No. of clusters')
plt.ylabel('Cost')
plt.title('Elbow Method For Optimal k')
plt.show()
```



We can see a bend at K=3 in the above graph indicating 3is the optimal number of clusters.

Build a model with 3 clusters

```
# Building the model with 3 clusters
kmode = KModes(n_clusters=3, init = "random", n_init = 5, verbose=1)
clusters = kmode.fit_predict(data)
clusters
```

Finally, insert the predicted cluster values in our original dataset.

```
data.insert(0, "Cluster", clusters, True)
data
```

:

person	Cluster	hair_color	eye_color	skin_color
P1	0	blonde	amber	fair
P2	0	brunette	gray	brown
P3	2	red	green	brown
P4	1	black	hazel	brown
P5	0	brunette	amber	fair
P6	1	black	gray	brown
P7	2	red	green	fair
P8	1	black	hazel	fair

Dataset after inserting predicted cluster values

Inference from the model predictions: P1, P2, P5 are merged as a cluster; P3, P7 are merged; and P4, P6, P8 are merged.

The results of our theoretical approach are in line with the model predictions. 🙌

End Notes:

By the end of this article, we are familiar with the working and implementation of the KModes clustering algorithm. In the upcoming article, we will be learning the K-prototype algorithm.

References:

KModes algorithm

GitHub Repo link

I hope this blog helps understand the KModes clustering algorithm. Please give it a clap 👏. Happy learning !!😍

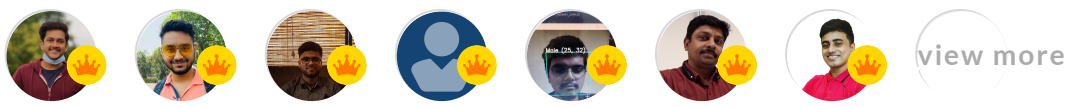
The media shown in this article are not owned by Analytics Vidhya and are used at the Author's discretion.

blogathon clustering machine learning python unsupervised learning

About the Author

Harika Bonthu

Our Top Authors



Previous Post

Predict Future Sales using XGBRegressor

Next Post

4 Ways to Handle Insufficient Data In Machine Learning!

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

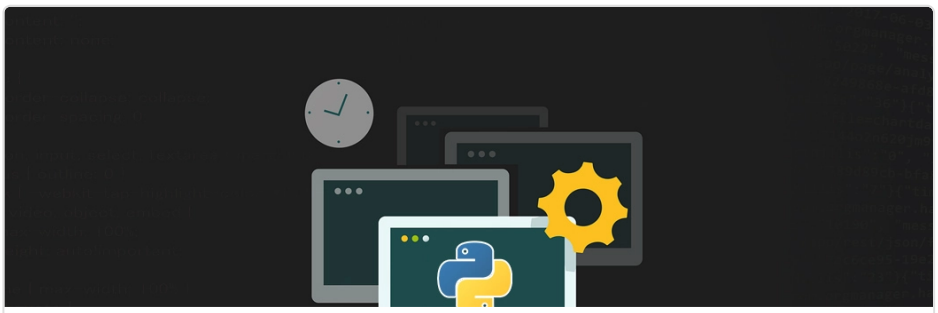
Name*

Email*

Website

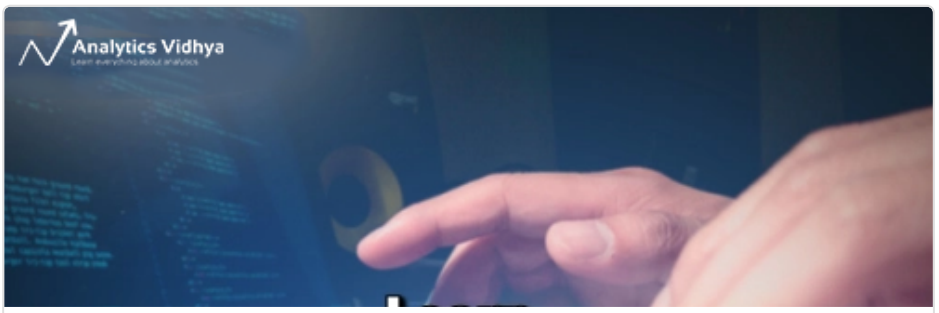
Submit

Top Resources



Python Tutorial: Working with CSV file for Data Science

Harika Bonthu - AUG 21, 2021



Understanding Support Vector Machine(SVM) algorithm from examples (along with code)

sunil - SEP 13, 2017

Understanding Random Forest

Sruthi E R - JUN 17, 2021

6 Easy Steps to Learn Naive Bayes Algorithm with codes..

sunil - SEP 11, 2017