خوشه بندی k میانگین (k-means Clustering)

مقدمه ای بر الگوریتم خوشه بندی K-means :

الگوریتم خوشه بندی K-means به محاسبه نقاط مرکزی می پردازد و این کار را تکرار می کند تا نقطه مرکزی بهینه را پیدا کند. این الگوریتم فرض می کند که تعداد خوشه ها مشخص شده است. همچنین به آن الگوریتم خوشه بندی flat نیز گفته می شود. تعداد خوشه های مشخص شده در داده توسط الگوریتم، با k در K-means نمایش داده می شوند.

در این الگوریتم، نقاط داده به گونه ای به یک خوشه تخصیص داده می شوند که مجموع مربع فاصله بین نقاط داده و نقطه مرکزی کمینه باشد. قابل درک است که هر چه تغییرات در خوشه ها کمتر باشد، نقاط داده مشابه تری را درون خوشه خواهیم داشت.

عملکرد الگوریتم K-Means : با کمک مراحل زیر می توان نحوه عملکرد الگوریتم خوشه بندی K-Means را فهمید.

مرحله ۱ – ابتدا، باید تعداد خوشه های(k) مورد نیاز جهت تولید توسط الگوریتم را مشخص کنیم.

مرحله ۲– سپس، به صورت تصادفی تعداد k نقطه داده را انتخاب کنید و هر یک را به یک خوشه تخصیص دهید. به بیان ساده تر، داده را بر اساس تعداد نقاط داده طبقه بندی کنید.

مرحله ۳ – حال الگوریتم، نقاط مرکزی خوشه را محاسبه خواهد کرد.

مرحله ۴– سپس، مراحل زیر را تکرار کنید تا نقطه مرکزی بهینه که تخصیص نقاط داده به خوشه هایی است که دیگر تغییر نمی کنند، را پیدا کنیم.

۴.۱ – ابتدا، مجموع مربع فاصله بین نقاط داده و نقاط مرکزی محاسبه خواهد شد.

۴.۲ – حال باید هر نقطه داده را به خوشه ای (نقطه مرکزی) که از سایر خوشه ها نزدیک تر است تخصیص دهیم.

۴.۳ – در انتها، با محاسبه میانگین همه نقاط داده آن خوشه، نقاط مرکزی را برای خوشه ها محاسبه کنید.

الگوریتم K-means از روش **Expectation-Maximization** برای حل مساله استفاده می کند. از مرحله انتظار (Expectation) برای تخصیص نقاط داده به نزدیکترین خوشه استفاده می شود و مرحله بیشینه سازی (Maximization) برای محاسبه نقطه مرکزی هر خوشه مورد استفاده قرار میگیرد.

در زمان کار با الگوریتم K-means، باید موارد زیر را در نظر داشته باشیم.

- در زمان کار با الگوریتم های خوشه بندی مانند K-means، توصیه می شود داده را به صورت استاندارد استفاده کنید، زیرا چنین الگوریتم هایی از اندازه گیری مبتنی بر فاصله برای تعیین شباهت بین نقاط داده استفاده می کنند.
- به علت ذات تکرار کننده K-Means و مقدار دهی اولیه نقاط مرکزی به صورت تصادفی، ممکن است K-Means به یک نقطه بهینه محلی برسد و به یک نقطه بهینه سراسری همگرا نشود.

پیاده سازی در پایتون: دو مثال زیر از پیاده سازی الگوریتم خوشه بندی K-Means در پایتون، ما را در درک بهتر آن کمک خواهد کرد.

مثال ۱ : این یک مثال ساده برای درک نحوه عملکرد K-Means است. در این مثال، در ابتدا مجموعه داده دو بعدی (2D) شامل ۴ blob مختلف را تولید میکنیم و سپس الگوریتم k-means را اعمال می کنیم تا نتیجه را ببینیم.

ابتدا، با وارد کردن بسته های ضروری شروع میکنیم.

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np
```

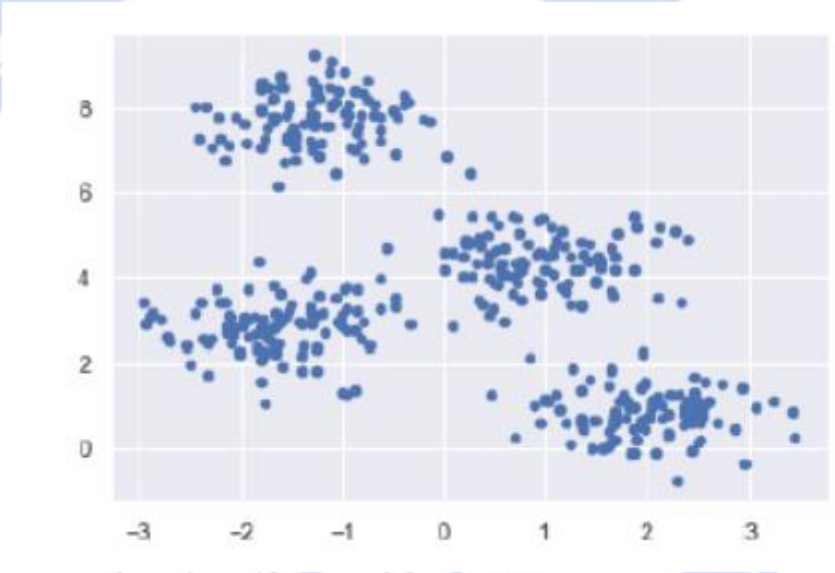
```
from sklearn.cluster import KMeans
```

کد زیر مجموعه داده 2D شامل 4 blob را تولید میکند.

```
from sklearn.datasets.samples_generator import make_blobs
X, y_true = make_blobs(n_samples = 400, centers = 4, cluster_std = 0.60, random_state = 0)
```

سپس، کد زیر در ترسیم مجموعه داده به ما کمک می کند.

```
plt.scatter(X[:, 0], X[:, 1], s = 20);
plt.show()
```



سپس، به همراه ارائه تعداد خوشه ها، یک شی از KMeans بسازید. مدل را آموزش دهید و به صورت زیر پیش بینی را انجام دهید.

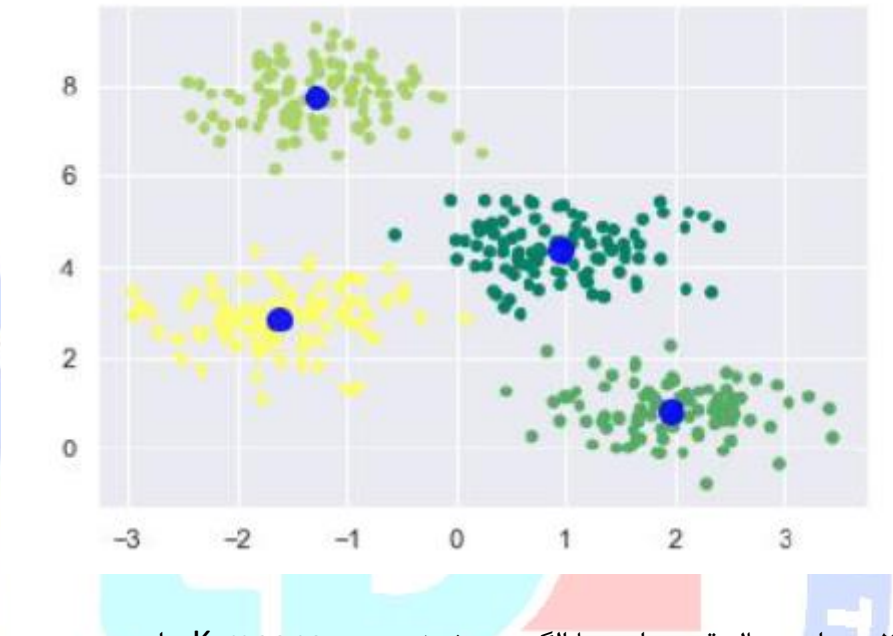
```
kmeans = KMeans(n_clusters = 4)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
```

حال، با کمک کد زیر می توانیم مراکز خوشه ها را که توسط برآوردگر k-means پایتون انتخاب شده است را رسم کنیم.

```
from sklearn.datasets.samples_generator import make_blobs
X, y_true = make_blobs(n_samples = 400, centers = 4, cluster_std = 0.60, random_state = 0)
```

سپس، کد زیر در تصویر سازی مجموعه داده به ما کمک خواهد کرد.

```
plt.scatter(X[:, 0], X[:, 1], c = y_kmeans, s = 20,
cmap = 'summer')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c = 'blue',
s = 100, alpha = 0.9);
plt.show()
```



مثال ۲ : در این مثال قصد داریم تا الگوریتم خوشه بندی K-means را روی مجموعه داده ارقام ساده اعمال کنیم. K-means تلاش خواهد کرد تا بدون استفاده از اطلاعات برچسب اصلی، ارقام مشابه را شناسایی کند.

ابتدا، با وارد کردن بسته های ضروری شروع خواهیم کرد.

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np
from sklearn.cluster import KMeans
```

سپس، مجموعه داده ارقام را از sklearn بارگیری کنید و یک شی از آن بسازید. همچنین می توانیم به صورت زیر تعداد سطر ها و ستون های درون این مجموعه داده را نیز پیدا کنیم.

```
from sklearn.datasets import load_digits
digits = load_digits()
digits.data.shape
```

خروجی:

```
(1797, 64)
```

خروجی فوق نشان می دهد که این مجموعه داده دارای ۱۷۹۷ نمونه با ۶۴ ویژگی است. به صورت مشابه در مثال ۱ ، خوشه بندی را اعمال می کنیم.

```
kmeans = KMeans(n_clusters = 10, random_state = 0)
clusters = kmeans.fit_predict(digits.data)
kmeans.cluster_centers_.shape
```

خروجی:

```
(10, 64)
```

خروجی فوق نشان می دهد که K-means ۱۰ خوشه با ۶۴ ویژگی ساخته است.

```
fig, ax = plt.subplots(2, 5, figsize=(8, 3))
centers = kmeans.cluster_centers_.reshape(10, 8, 8)
for axi, center in zip(ax.flat, centers):
axi.set(xticks=[], yticks=[])
axi.imshow(center,          interpolation='nearest',
cmap=plt.cm.binary)
```

خروجی: به عنوان خروجی، تصویر زیر را نشان می دهیم که مراکز خوشه ها را با k-means آموخته است.

کد زیر برچسب های خوشه آموخته شده را با برچسب های واقعی موجود در آنها تطابق می دهد.

```
from scipy.stats import mode
labels = np.zeros_like(clusters)
for i in range(10):
   mask = (clusters == i)
   labels[mask] = mode(digits.target[mask])[0]
```

سپس، به صورت زیر می توانیم دقت را بررسی کنیم.

```
from sklearn.metrics import accuracy_score
accuracy_score(digits.target, labels)
```

خروجی:

```
0.7935447968836951
```

خروجی فوق نشان می دهد که دقت تقریبا ۸۰ درصد است.

مزایا و معایب:

مزایا: موارد زیر برخی از مزایای الگوریتم های خوشه بندی K-Means است: فهم و پیاده سازی آن بسیار آسان است. اگر تعداد متغیر های بسیاری داشته باشیم، K-Means از خوشه بندی سلسله مراتبی سریع تر خواهد بود. در محاسبه مجدد نقاط مرکزی، یک نمونه می تواند خوشه را تغییر دهد. در مقایسه با خوشه بندی سلسله مراتبی، K-Means خوشه های محکم تری را شکل می دهد.

معایب: موارد زیر برخی از معایب الگوریتم های خوشه بندی K-Means است: پیش بینی تعداد خوشه ها (مقدار k) کمی مشکل است. خروجی به شدت تحت تاثیر ورودی اولیه، مانند تعداد خوشه ها (مقدار k) است. ترتیب داده ها تاثیر قوی روی خروجی نهایی دارد. نسبت به بازسازی مجدد بسیار حساس است، اگر با استفاده از نرمال سازی یا استاندارد سازی داده های خود را مجددا بازسازی کنیم، خروجی کاملا تغییر خواهد کرد. اگر خوشه ها یک شکل هندسی پیچیده داشته باشند، خوشه بندی کار مناسبی نخواهد بود.

برنامه های کاربردی الگوریتم خوشه بندی K-means: اهداف اصلی از تحلیل خوشه عبارتند از:

۱- دریافت یک درک معنادار از داده ای که با آن کار می کنیم. ۲- خوشه بندی و سپس پیش بینی در جایی که مدل های مختلف برای زیر گروه های مختلف ساخته خواهد شد.

برای تحقق اهداف فوق، الگوریتم خوشه بندی K-means به خوبی عمل می کند. می توان از آن در برنامه های کاربردی زیر استفاده کرد.

تفکیک بازار . خوشه بندی اسناد. تفکیک تصویر . فشرده سازی تصویر. تفکیک مشتری. تحلیل روند روی داده پویا.