

دانشگاه صنعتي امیرکبیر  
(پلی تکنیک تهران)

دانشكده مهندسی برق

ﮔﺰﺍﺭﺵکار ﺁﺯﻣﺎﻳﺶ چهارم

ﺁﺯﻣﺎﻳﺸﮕﺎﻩ مقدمه ای بر ﻫﻮﺵ ﻣﺤﺎﺳﺒﺎﺗﻲ

پیاده سازی K-Means Clustering

نگارش

ارشیا اسمعیل طهرانی

علی بابالو

پویا ابراهیمی

استاد راهنما

سرکار خانم موسوی

آذر ماه 1401

# چکیده

در این آزمایش به پیاده سازیK-Means Clustering پرداخته شده است.

|  |  |
| --- | --- |
| فهرست مطالب | صفحه |

**پیش گزارش....................................................................................................................................1**

مفهوم الگوريتم با نظارت و بدون نظارت....................................................................................................1

بیان مزيت هاي الگوريتم هاي با نظارت نسبت به الگوريتم هاي بدون نظارت و بر عكس.............1

توضیح الگوریتم دسته بندی K – Means..............................................................................................3

کاربرد های الگوریتم K – Means ..........................................................................................................4

**شرح آزمایش..................................................................................................................................6**

محیط Python..............................................................................................................................................7

کلاس KMeans............................................................................................................................7

Train کردن مدل.........................................................................................................................13

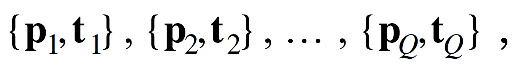
ارزیابی الگوریتم.........................................................................................................................................14

کاهش حجم عکس با استفاده از K-Means...................................................................................16

**تمارین..............................................................................................................................18**

# پیش گزارش

## مفهوم الگوريتم با نظارت و بدون نظارت

در یادگیری نظارت شده، قانون یادگیری با مجموعه‌ای از مثال‌ها (مجموعه آموزشی) رفتار مناسب شبکه ارائه می‌شود:

که در آن یک ورودی به شبکه و خروجی صحیح (هدف) مربوطه است. همانطور که ورودی ها به شبکه اعمال می شوند، خروجی های شبکه با اهداف مقایسه می شوند. سپس از قانون یادگیری برای تنظیم وزن ها و بایاس های شبکه استفاده می شود تا خروجی های شبکه به اهداف نزدیک تر شود.

اما در یادگیری بدون نظارت، وزن ها و Bias ها فقط در پاسخ به ورودی های شبکه اصلاح می شوند. هیچ خروجی هدفی در دسترس نیست. در نگاه اول ممکن است این غیرعملی به نظر برسد. اگر نمی دانید که قرار است چه کاری انجام دهد، چگونه می توانید یک شبکه را آموزش دهید؟ اکثر این الگوریتم ها نوعی عملیات خوشه بندی را انجام می دهند. آنها یاد می گیرند که الگوهای ورودی را در تعداد محدودی از کلاس ها طبقه بندی کنند. این به ویژه در کاربردهایی مانند کوانتیزاسیون برداری مفید است.

## بیان مزيت هاي الگوريتم هاي با نظارت نسبت به الگوريتم هاي بدون نظارت و بر عكس

### مزایایSupervised learning

* این به شما امکان می دهد در مورد تعریف برچسب ها بسیار دقیق باشید. به عبارت دیگر، می‌توانید الگوریتم را برای تشخیص کلاس‌های مختلف آموزش دهید، جایی که می‌توانید یک مرز تصمیم‌گیری ایده‌آل تعیین کنید.
* شما می توانید تعداد کلاس هایی را که می خواهید داشته باشید تعیین کنید.
* داده های ورودی بسیار شناخته شده و دارای برچسب هستند.
* نتایج حاصل از روش نظارت شده در مقایسه با نتایج حاصل از تکنیک‌های بدون نظارت یادگیری ماشین دقیق‌تر و قابل اعتمادتر هستند. این عمدتا به این دلیل است که داده های ورودی در الگوریتم نظارت شده به خوبی شناخته شده و دارای برچسب هستند. این یک تفاوت کلیدی بین یادگیری تحت نظارت و بدون نظارت است.
* پاسخ‌های موجود در تجزیه و تحلیل و خروجی الگوریتم شما احتمالاً مشخص است، زیرا همه کلاس‌های استفاده شده شناخته شده هستند.

### معایب Supervised learning

* یادگیری تحت نظارت در مقایسه با روش بدون نظارت می تواند روشی پیچیده باشد. دلیل اصلی این است که شما باید به خوبی درک کنید و ورودی ها را در یادگیری نظارت شده برچسب گذاری کنید.
* این در Real – time انجام نمی شود در حالی که یادگیری بدون نظارت در مورد Real – Time است. این نیز یک تفاوت عمده بین یادگیری تحت نظارت و بدون نظارت است. یادگیری ماشین نظارت شده از تجزیه و تحلیل خطی استفاده می کند.
* برای آموزش به زمان محاسباتی زیادی نیاز است.
* اگر داده‌های بزرگ و در حال رشد پویا وجود دارد، و برچسب‌هایی که قوانین را از قبل تعریف کرده اند غیر قابل اطمینان اند. این می تواند یک چالش واقعی باشد.

### مزایای Unsupervised learning

* پیچیدگی کمتر در مقایسه با یادگیری تحت نظارت. برخلاف الگوریتم های نظارت شده، در یادگیری بدون نظارت، هیچ کس نیازی به درک و سپس برچسب گذاری ورودی داده ها ندارد. این امر یادگیری بدون نظارت را پیچیده تر می کند و توضیح می دهد که چرا بسیاری از افراد تکنیک های بدون نظارت را ترجیح می دهند.
* در زمان واقعی انجام می شود به طوری که تمام داده های ورودی در حضور فراگیران تجزیه و تحلیل و برچسب گذاری می شود. این به آنها کمک می کند تا مدل های مختلف یادگیری و مرتب سازی داده های خام را به خوبی درک کنند.
* دریافت داده‌های بدون برچسب اغلب آسان‌تر است - از رایانه نسبت به داده‌های برچسب‌گذاری شده، که نیاز به مداخله شخص دارند. این نیز یک تفاوت کلیدی بین یادگیری تحت نظارت و بدون نظارت است.

### معایب Unsupervised learning

* شما نمی توانید در مورد تعریف مرتب سازی داده ها و خروجی خیلی دقیق صحبت کنید. این به این دلیل است که داده های مورد استفاده در یادگیری بدون نظارت برچسب گذاری شده و شناخته شده نیستند. این وظیفه ماشین است که داده های خام را قبل از تعیین الگوهای پنهان برچسب گذاری و گروه بندی کند.
* دقت کمتر نتایج این نیز به این دلیل است که داده های ورودی از قبل توسط افراد مشخص نیست و برچسب گذاری نشده است، به این معنی که ماشین باید این کار را به تنهایی انجام دهد.
* نتایج تجزیه و تحلیل را نمی توان مشخص کرد. هیچ دانش قبلی در مورد روش بدون نظارت یادگیری ماشین وجود ندارد. علاوه بر این، تعداد کلاس ها نیز مشخص نیست. منجر به عدم توانایی در تعیین نتایج حاصل از تجزیه و تحلیل می شود.

## توضیح الگوریتم دسته بندی K – Means

الگوریتم چگونه کار می کند؟ ، فرض کنید centroid ها به شما داده شده است. شما به راحتی می توانید تمام نمونه های موجود در مجموعه داده را با اختصاص دادن هر یک از آنها به خوشه ای که مرکز آن نزدیک ترین است، برچسب گذاری کنید. برعکس، اگر همه برچسب‌های نمونه به شما داده شود، می‌توانید به راحتی مرکز هر خوشه را با محاسبه میانگین نمونه‌های آن خوشه پیدا کنید. اما به شما نه برچسب و نه مرکز داده شده است، پس چگونه می توانید ادامه دهید؟

* با قرار دادن سانتروئیدها به طور تصادفی شروع کنید (به عنوان مثال، با انتخاب k نمونه به طور تصادفی از مجموعه داده و استفاده از مکان آنها به عنوان مرکز).
* سپس نمونه‌ها را برچسب بزنید، مرکزها را به‌روزرسانی کنید،
* نمونه‌ها را برچسب بزنید، مرکزها را به‌روزرسانی کنید،
* و به همین ترتیب تا زمانی که مرکزها از حرکت بازایستند.

این الگوریتم تضمین شده است که در تعداد محدودی از مراحل (معمولاً بسیار کوچک) همگرا شود. این به این دلیل است که میانگین مجذور فاصله بین نمونه‌ها و نزدیک‌ترین مرکز آنها فقط می‌تواند در هر مرحله پایین بیاید و از آنجایی که نمی‌تواند منفی باشد، تضمین می‌شود که همگرا شود.

اگرچه الگوریتم تضمین شده است که همگرا شود، اما ممکن است به راه حل مناسب همگرا نشود (یعنی ممکن است به یک بهینه محلی همگرا شود): اینکه آیا این الگوریتم به مقدار دهی اولیه مرکز بستگی دارد.

ﺍﻟﮕﻮﺭﻳﺘﻢ K – Means ﻳﻚ ﺍﻟﮕﻮﺭﻳﺘﻢ ﺑﺎﺯﮔﺸﺘﻲ ﺍﺳﺖ ﻛﻪ ﺑﺎ ﻳﻚ ﻓﺮﺽ ﺍﻭﻟﻴﻪ ﺩﺭﺑﺎﺭﻩ ﻣﺮﺍﻛﺰ ﺩﺳﺘﻪ ﻫﺎ ﺁﻏﺎﺯ ﻣﻲ ﺷﻮﺩ ﺩﺭ ﻫﺮ ﻣﺮﺣﻠﻪ ﺍﺯ ﺍﺟﺮﺍﻱ ﺍﻟﮕﻮﺭﻳﺘﻢ ﻣﺮﺍﺣﻞ ﺯﻳﺮ ﺍﺟﺮﺍ ﻣﻲ ﺷﻮﺩ:

1. ﭘﻴﺪﺍ ﻛﺮﺩﻥ ﺩﺳﺘﻪ ﻣﺘﻨﺎﻇﺮ ﺑﺎ ﺗﻤﺎﻡ ﻧﻘﺎﻁ
2. ﺗﺎﺯﻩ ﺳﺎﺯﻱ ﻣﺮﺍﻛﺰ ﺩﺳﺘﻪ ﻫﺎ

ﺩﺭ ﻣﺮﺣﻠﻪ ﺍﻭﻝ ﺍﺟﺮﺍﻱ ﺍﻟﮕﻮﺭﻳﺘﻢ ﻓﺎﺻﻠﻪ ﺗﻤﺎﻡ ﻧﻘﺎﻁ ﺗﺎ ﻣﺮﺍﻛﺰ ﺩﺳﺘﻪ ﻫﺎ ﻣﺤﺎﺳﺒﻪ ﻣﻲ ﺷﻮﺩ ﻭ ﺳﭙﺲ ﻫﺮ ﺩﺍﺩﻩ ﻣﺘﻌﻠﻖ ﺑﻪ ﺩﺳﺘﻪ ﺍﻱ ﻛﻪ ﻛﻤﺘﺮﻳﻦ ﻓﺎﺻﻠﻪ ﺑﺎ ﺁﻥ ﺭﺍ ﺩﺍﺭﺩ ﻣﻲ ﺷﻮﺩ.

ﺑﻌﺪ ﺍﺯ ﭘﻴﺪﺍ ﻛﺮﺩﻥ ﺩﺳﺘﻪ ﻫﺎﻱ ﻣﺘﻨﺎﻇﺮ ﺑﺎ ﻫﺮ ﺩﺍﺩﻩ ، ﺩﺭ ﻣﺮﺣﻠﻪ ﺩﻭﻡ ﻣﻴﺎﻧﮕﻴﻦ ﺩﺍﺩﻩ ﻫﺎﻱ ﻣﺘﻌﻠﻖ ﺑﻪ ﻳﻚ ﺩﺳﺘﻪ ﺑﻪ ﻋﻨﻮﺍﻥ ﻣﺮﻛﺰ ﺩﺳﺘﻪ ﺩﺭ ﻧﻈﺮ ﮔﺮﻓﺘﻪ ﻣﻲ ﺷﻮﺩ.

ﻣﺮﺍﺣﻞ۱ ﻭ۲ ﺗﺎ ﺯﻣﺎﻧﻲ ﻛﻪ ﻣﺮﺍﻛﺰ ﺩﺳﺘﻪ ﻫﺎ ﺗﻐﻴﻴﺮ ﻧﻜﻨﺪ ﻭ ﻳﺎ ﺗﻐﻴﻴﺮﺍﺕ ﺧﻴﻠﻲ ﻛﻤﻲ ﺩﺍﺷﺘﻪ ﺑﺎﺷﻪ ﺍﺩﺍﻣﻪ ﻣﻲ ﻳﺎﺑﺪ.

ﺑﻌﺪ ﺍﺯ ﺛﺎﺑﺖ ﺷﺪﻥ ﻣﺮﺍﻛﺰ ﺩﺳﺘﻪ ﻫﺎ ﺍﻟﮕﻮﺭﻳﺘﻢ ﻫﻤﮕﺮﺍ ﻣﻲ ﺷﻮﺩ.

## کاربرد های الگوریتم K – Means

خوشه بندی در طیف گسترده ای از برنامه ها استفاده می شود، از جمله:

### تقسیم بندی مشتریان

شما می توانید مشتریان خود را بر اساس خرید و فعالیت آنها در وب سایت خود دسته بندی کنید. این برای درک اینکه مشتریان شما چه کسانی هستند و به چه چیزی نیاز دارند مفید است، بنابراین می توانید محصولات و کمپین های بازاریابی خود را با هر بخش تطبیق دهید. برای مثال، تقسیم‌بندی مشتری می‌تواند در سیستم‌های توصیه‌گر برای پیشنهاد محتوایی که سایر کاربران در همان خوشه از آن لذت می‌برند، مفید باشد.

### تحلیل داده ها

هنگامی که یک مجموعه داده جدید را تجزیه و تحلیل می کنید، اجرای یک الگوریتم خوشه بندی و سپس تجزیه و تحلیل هر خوشه به طور جداگانه می تواند مفید باشد.

### کاهش ابعاد

هنگامی که یک مجموعه داده خوشه‌بندی شد، معمولاً می‌توان وابستگی هر نمونه را با هر خوشه اندازه‌گیری کرد. وابستگی هر معیاری است که نشان می دهد یک نمونه چقدر در یک خوشه قرار می گیرد. سپس بردار ویژگی هر نمونه x را می توان با بردار قرابت های خوشه ای آن جایگزین کرد. اگر k خوشه وجود داشته باشد، این بردار k بعدی است. بردار جدید معمولاً ابعاد بسیار کمتری نسبت به بردار ویژگی اصلی دارد، اما می تواند اطلاعات کافی را برای پردازش بیشتر حفظ کند.

### مهندسی ویژگی

پیوندهای خوشه اغلب می توانند به عنوان ویژگی های اضافی مفید باشند. به عنوان مثال، K - Means برای افزودن ویژگی‌های وابسته به خوشه جغرافیایی به مجموعه داده مسکن کالیفرنیا استفاده می‌شود، و آنها به عملکرد بهتر کمک کردند.

### تشخیص ناهنجاری (Outliers)

هر نمونه ای که تمایل کمی به همه خوشه ها داشته باشد، احتمالاً یک ناهنجاری است. به عنوان مثال، اگر کاربران وب سایت خود را بر اساس رفتار آنها دسته بندی کرده باشید، می توانید کاربرانی را که رفتار غیرعادی دارند، مانند تعداد غیرعادی درخواست در ثانیه شناسایی کنید.

### یادگیری نیمه نظارتی

اگر فقط چند برچسب دارید، می‌توانید خوشه‌بندی را انجام دهید و برچسب‌ها را در همه نمونه‌های یک خوشه منتشر کنید. این تکنیک می‌تواند تعداد برچسب‌های موجود برای الگوریتم یادگیری تحت نظارت بعدی را تا حد زیادی افزایش دهد و در نتیجه عملکرد آن را بهبود بخشد.

### موتورهای جستجو

برخی از موتورهای جستجو به شما امکان می دهند تصاویری را جستجو کنید که شبیه به یک تصویر مرجع هستند. برای ساختن چنین سیستمی، ابتدا باید یک الگوریتم خوشه بندی را برای تمام تصاویر موجود در پایگاه داده خود اعمال کنید. تصاویر مشابه به یک خوشه ختم می شوند. سپس هنگامی که یک کاربر یک تصویر مرجع ارائه می دهد، تنها کاری که باید انجام دهید این است که از مدل خوشه بندی آموزش دیده برای یافتن خوشه این تصویر استفاده کنید و سپس می توانید به سادگی تمام تصاویر را از این خوشه برگردانید.

### تقسیم بندی تصویر

با خوشه بندی پیکسل ها بر اساس رنگ آنها و سپس جایگزینی رنگ هر پیکسل با رنگ متوسط خوشه آن، می توان تعداد رنگ های مختلف در یک تصویر را به میزان قابل توجهی کاهش داد. بخش‌بندی تصویر در بسیاری از سیستم‌های تشخیص و ردیابی اشیا استفاده می‌شود، زیرا تشخیص کانتور هر شی را آسان‌تر می‌کند.

# شرح آزمایش

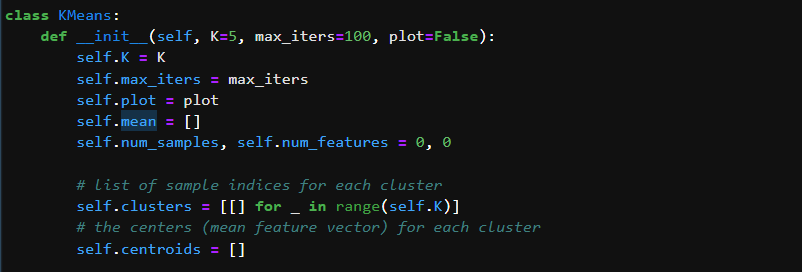
در این قسمت به پیاده سازی K – Means در محیط پایتون نیز پرداخته شده است.

## محیط Python

### کلاس KMeans

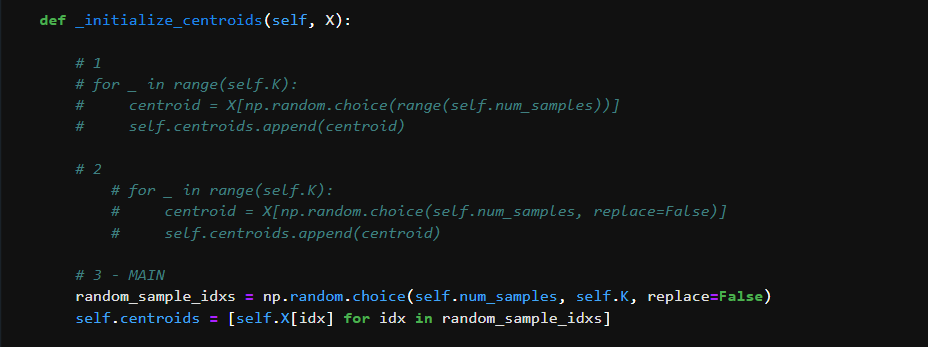
در ابتدای امر کتابخانه های مورد نیاز را import می‌کنیم. و seed برای random را یک مقدار دلخواه انتخاب می‌کنیم تا با هر بار اجرای برنامه مقادیر مختلف random نیز گرفته نشود!

شکل۱) ایمپورت کردن کتابخانه ها

در ابتدا در تابع کانستراکتور یا سازنده کلاس مقادیر مورد نیاز را مقدار دهی اولیه می‌کنیم؛

شکل۲) کانستراکتور کلاس K-Means

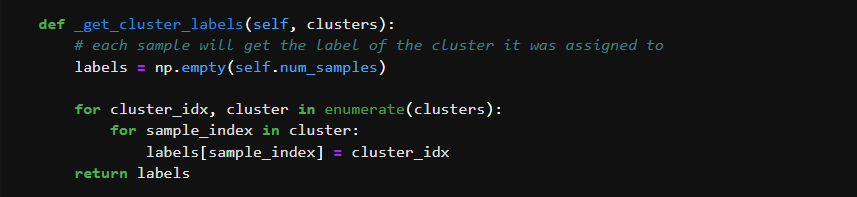
* K برای تعداد کلاستر های ورودی
* Max\_iters برای حداکثر iteration هایی که طی شود
* Plot برای اینکه آیا step by step چاپ بکند یا خیر
* Mean لیستی است که در آن ها میانگین فاصله داده های از centroid محاسبه شده و ریخته می‌شود.
* Num\_samples و num\_features به ترتیب شماره sample ها و شماره feature ها هستند.
* Clusters، لیستی از لیست ها است که در آن اندیس های عضور های داخل هر یک از کلاستر ها، داخل هر یک از المان های آن ریخته می‌شود که تعداد المان های آن، پرواضح است که برابر با تعداد کلاستر های ‌می‌باشد.
* Centroids لیستی است که در آن centroid ها ریخته می‌شود.

سپس به مقدار دهی کردن اولیه centroid هایمان می‌پردازیم؛

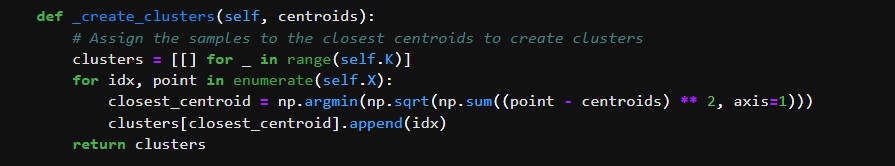
شکل۳) مقدار دهی اولیه به مراکز دسته ها

* روش اول و دوم؛ پس از پیاده سازی این روش ها که در کد آورده شده است، مشخص شد که در خروجی برخی از میانگین هایمان نیز Nan ظاهر می‌شود. پس از بررسی فراوان مشخص شد که دلیل این امر، این بود که چون در هر iteration حلقه، از sample ها به صورت random حتی با وجود غیر تکراری بودن، نمونه برداری می‌شود، و با نمونه برداری در هر حلقه، بررسی نمی‌شود که آیا مقدار برداشته شده، با مقادیر برداشته شده در حلقه های قبل یکسان می‌باشد یا خیر، آن گاه در برخی از k ها و مخصوصا در k های بالاتر نیز، مقادیر تکراری برای centroid ها وجود می‌داشت و وجود این مقادیر تکراری معادلات و محاسبات را بر هم می‌زد و باعث ظهور مقادیر Nan در میانگین ها می‌شد!
* روش سوم؛ بنابراین تصمیم بر این شد که ابتدا و فقط یک بار بدون تکرار مقادیر تصادفی ای را از num\_sample هایمان انتخاب و نمونه برداری کنیم(اندیس های آن ها انتخاب می‌شوند)، و سپس، با یک list comprehension، مقادیر centroid ها را از آن ها برداریم!

در ادامه در متد \_get\_cluster\_labels، هر sample ای، اندیس کلاستری را که عضو آن است، می‌گیرد!

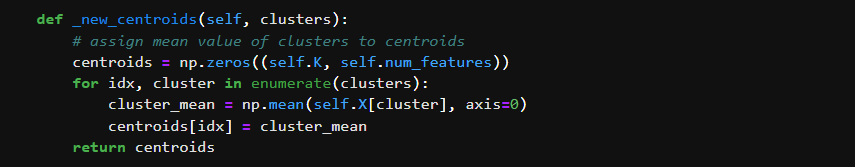
شکل۴) دسته بندی کلاستر ها

* توجه شود که np.empty، یک array جدیدی را با type و shape ورودی آن یعنی num\_samples را می‌گیرد!
* در کلاستر های enumerate می‌کند، همچنین در سمپل های هر کلاستر می‌چرخد، و سمپل مخصوص هر کلاستر را، اندیس آن کلاستر را بهش می‌دهد، و در نهایت label ها نیز return می‌شوند!

سپس در \_create\_clusters، لیستی از لیست ها که لیست های داخل به تعداد k هستند، ایجاد شده؛

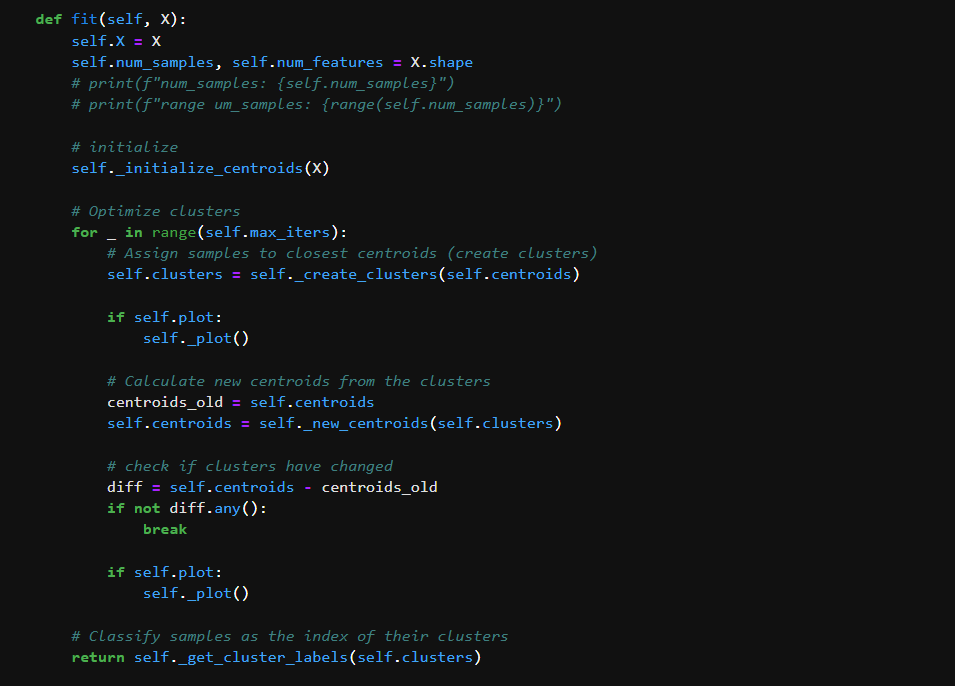
شکل۵) تابع برای دسته بندی کردن ورودی

* فاصله هر point را با مراکز کلاستر ها مقایسه می‌کند و کمترین آن فاصله را پیدا کرده. سپس، اندیس آن point را در عضو clusters که اندیس آن فاصله min را دارد، افزوده و در نهایت clusters را return می‌کند!

در متد \_new\_centroids، میانگین داده های عضو هر کلاستر محاسبه و مشخص می‌شود که کدام عضو است، و centroid جدید را برابر آن عضو قرار می‌دهد.

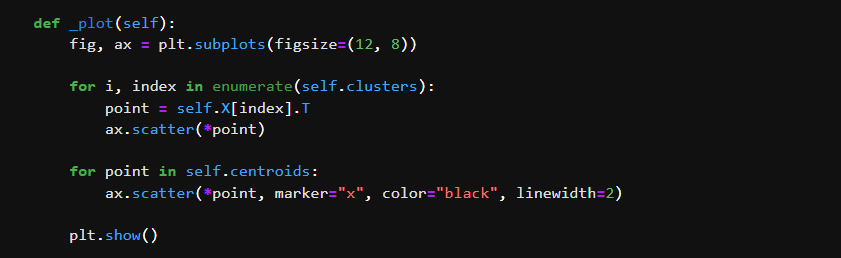
شکل۶) تابع برای تغییر مراکز

* به این گونه عمل می‌کند که ابتدا آرایه ای جدید با shape و type آرگومان داخلی اش تولید می‌شود که تعداد سطر های آن k یا تعداد کلاستر ها، و تعداد ستون های آن تعداد feature ها می‌باشد.
* در clusters، enumerate می‌کند که با این کار اندیس هر عضو هم در می‌آید، میانگین داده های هر کلاستر به صورت سطری نیز حساب می‌شود و مقدار لیبلcentroid، برابر با مقداد label آن اندیس نیز قرار می‌گیرد.

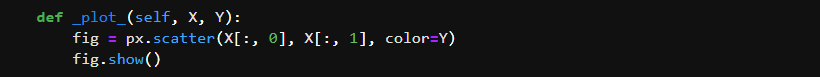
در ادامه در متد fit، مقادیر X، Num\_samples، Num\_features، مقدار دهی می‌شوند،

شکل۷) predict/fit کردن ورودی

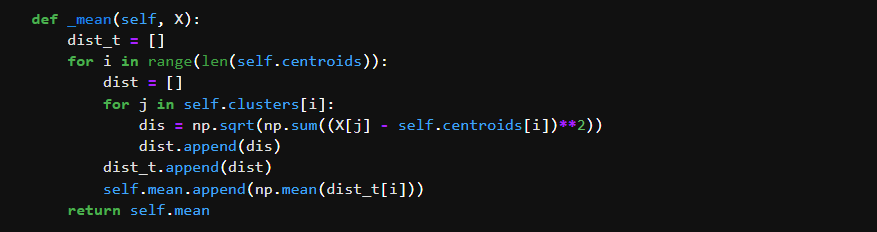
* با فراخوانی متد \_initialize\_centroids، centroid ها نیز مقدار دهی اولیه می‌شوند.
* در loop نیز با هر بار epoch، به واسطه ی centroid های مرحله قبلی و یک سری operation، کلاستر های ساخته می‌شوند.
* کلاستر های با \_creat\_cluster نیز ساخته شده؛
* اگر آرگومان plot، True بود، در هر epoch، نمودار های plot می‌شوند.
* Centroids\_old و centroids های جدید نیز مقدار دهی می‌شوند،
* در diff نیز اختلاف آن ها محاسبه می‌شود،
* اگر همه آن ها 0 شده بود، حلقه نیز break می‌کند،
* در نهایت، سمپل ها بر اساس اندیس کلاسترشان، دسته بندی و مشخص می‌شوند و در نهایت return می‌شوند.

در متد \_plot نیز محل قرار گیری point ها و centroid هر کلاستر نیز رسم می‌شود، و اگر flag مربوط به plot در آرگومان ورودی کلاس نیز True باشد، در هر حلقه نیز رسم می‌شود؛

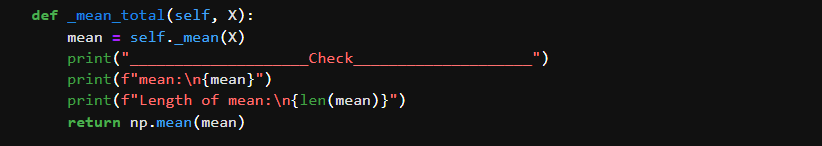
شکل۸) پلات کردن نقطه ها و نشان دادن مرکز هر دسته

شکل۹) در متد \_plot\_ به نحوی دیگر داده ها نمایش داده می‌شوند

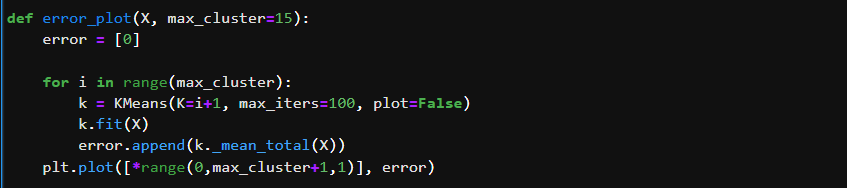
در ادامه، تابع ذیل، نقاط و دسته بندي متناظر با آن ها را از ورودي دریافت می‌کند، سپس براي هر دسته ميانگين فاصله نقاط متعلق به دسته مورد نظر را تا مركز دسته محاسبه می‌كند که به اين عدد خطاي هر دسته گفته مي شود.

شکل۱۰) محاسبه میانگین فاصله هر نقطه از مرکز دسته اش

* در dist، میانگین فاصله نقاط هر کلاستر تا مرکز آن کلاستر مشخص شده است،
* در dist\_t، dist ها ذخیره شده است
* ابتدا در dis نیز فاصله هر نقطه تا مرکز مشخص شده، سپس در dist نیز append می‌شود و در نهایت dist در dist\_t نیز append می‌شود. و در آخر میانگین dist\_t ها در mean نیز append شده و mean بازگردانده می‌شود.

در متد \_mean\_total نیز، متد \_mean فراخوانی می‌شود و در نهایت با میانگین گیری از خروجی آن یعنی mean(mean) نیز، ميانگين خطاي دسته ها به عنوان خطاي الگوريتم محاسبه و به عنوان خروجي باز گردانده می‌شود.

شکل۱۱) محاسبه خطای نهایی که برابر میانگین همه میانگین فواصل از مرکز دسته است

در نهایت در متد error\_plot، میزان خطای clustering را به ازای تعداد دسته ها یا تعداد کلاستر های مختلف که از 1 تا 15 می‌باشد، محاسبه می‌شود، در اصل الگوریتم کلاسترینگ به ازای k های محتلف تا 15 پیاده سازی و اجرا می‌شود و خطای آن ها نیز در لیستی ذخیر شده و در نهایت plot می‌شوند.

شکل۱۲) پلات کردن خطا بر حسب k

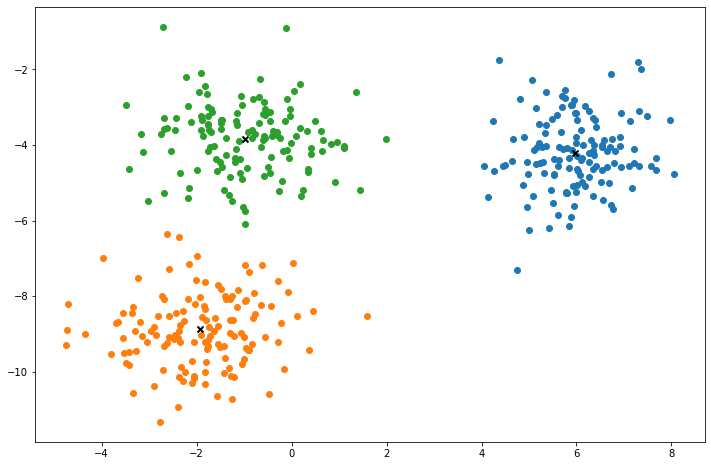
* عضو اول لیست error با 0 پر شده است چرا که برای plot شدن، مقادیر ارور باید با مقدار 0 جمع شوند و الگوریتم پیش برود.
* سپس در حلقه ای برای هر iteration و برای هر k ای نیز KMeans اجرا شده و خطای آن الگوریتم که پیش تر توسط متد \_mean\_total نیز محاسبه شده بود، به لیست error نیز append شده و در نهایت در span مشخصی نیز plot می‌شود.

**Train کردن مدل**

Graphical user interface, text, application

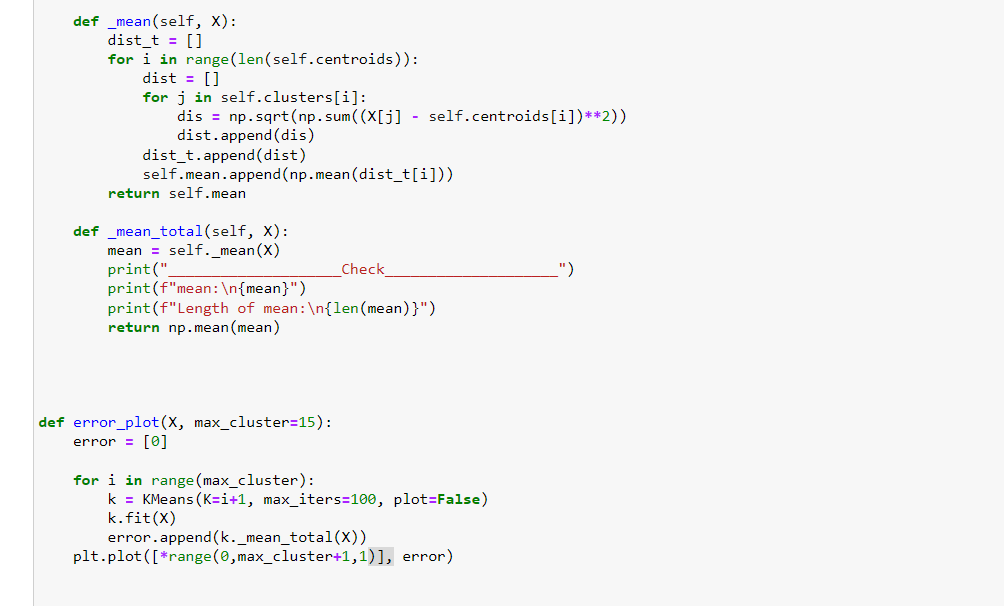
Description automatically generated

شکل۱۳)‌ اموزش الگوریتم Kmeans

برای آموزش الگوریتم K-Means از تابع make\_blobs در کتابخانه sklearn استفاده می­کنیم که شبکه را با استفاده از این دیتاست Train می­کنیم. که در شکل پایین خروجی الگوریتم را مشاهده می­کنیم که به درستی ۳ مرکز دسته هارا پیدا کرده است.

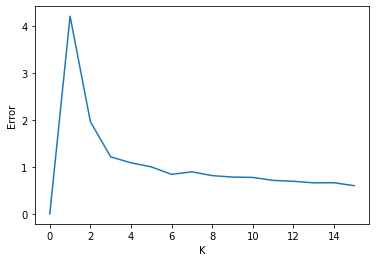
شکل۱۴) خروجی الگوریتم Kmeans

**ارزیابی الگوریتم**

شکل ۱۵) ارزیابی الگوریتم

برای ارزیابی الگوریتم یک تابع error\_plot تعریف میکنیم که از دوتابع \_mean, \_mean\_total استفاده می­کند که اولی میانگین فاصله نقاط کلاستر از مرکز آن را حساب می­کند و دیگری میانگین همه میانگین هارا حساب می­کند.

در تابع error\_plot یک لیست error تعریف میکنیم که ایندکس اول آن برابر ۰ است زیرا الگوریتم kmeans برای k = 0 تعریف نشده است پس برای پلات کردن آن از نقطه (۰-۰) شروع می­کنیم. این تابع یک ورودی max\_cluster میگیرد که ماکسیمم تعداد کلاستر های می­خواهیم برسی کنیم می­باشد. سپس یک حلقه در رنج ماکسیمم تعداد تشکیل می­دهیم که در هر حلقه کلاس kmeans را با K های مختلف ایجاد می­کند و دیتا را بر روی آن fit می­کند سپس ارور الگوریتم را بر لیست error، append میکند. در نهایت خطا را بر اساس مقدار k ( تعداد دسته ها) پلات می­کنیم که در تصویر پایین خروجی این تابع مشخص است.

شکل۱۶) نمودار خطا برحسب k

**کاهش حجم عکس با استفاده از K-Means:**

همانطور که در دستور کار توضیح داده شده است، یکی از کاربرد های این الگوریتم کاهش حجم عکس است گه در ادامه آن را توضیح خواهیم داد.

شکل ۱۷) تصویر مربوط به

در ابتدا عکس را به وسیله تابع imread() در کتابخانه OpenCv می­خوانیم که این تابع یک ماتریس به ابعاد ( طول\*عرض\*۳ ) میدهد که عدد ۳ مربوط به RGB هر پیکسل می­باشد. الگوریتم K-Means ما یک ورودی ۲ بعدی نیاز دارد که برای اینکه ما ماتریس ۳بعدیمان را به ۲ بعد تبدیل کنیم از تابع reshape استفاده میکنیم که سایز ماتریس را به تعداد کل پیکسل ها (طول\*عرض) \*‌۳ تبدیل بکند.

سپس یک کپی از ماتریسمان می­گیریم و آن را در ماتریس کامپرس شده ذخیره می­کنیم.

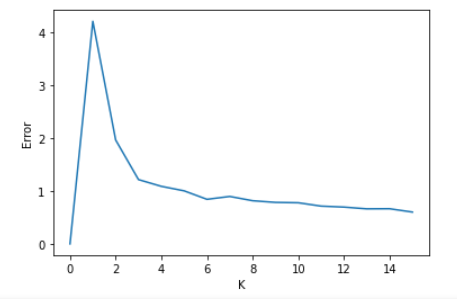
آنگاه کلاس Kmeans را با K دلخواه – مثلا ۱۶ – صدا می­کنیم و آن را با ماتریس ۲بعدیمان fit می­کنیم. سپس بعد از fit شدن دیتا هایمان، مقدار RGB های دیتا های هر کلاستر را برابر با مقدار RGB مرکز آن دسته قرار می­هیم و به این صورت حجم تصویرمان کاهش می­یابد، که این کار را در اینجا با استفاده از یک حلقه انجام دادیم. در انتها نیز این ماتریس ۲بعدی تصویر کاهش یافته را به ابعاد اصلی A person with a beard

Description automatically generated with low confidenceآن بر میگردانیم و با استفاده از تابع imwrite() ذخیره می­کنیم.

شکل ۱۸) تصویر اصلی و کاهش یافته با K = 2 از ارنست خالیموف ملقب به Giga-Chad

**تمارین**

۱ – همانطور که در نمودار خطا مشاهده کردیم با اضافه شدن مقادیر K بصورت میانگین مقدار خطا کاهش می­یابد اما بعد از حدی از K این کاهش خطا بسیار کم می­شود بصورتی که تقریبا تاثیری روی دقت الگوریتم ندارد و فقط حجم محاسبات را زیاد می­کند. برای پیدا کردن نقطه بهینه الگرویتم باید نقطه elbow خطا را پیدا کنیم و آن را مقدار بهینه K برای استفاده از الگوریتم بیان کنیم.

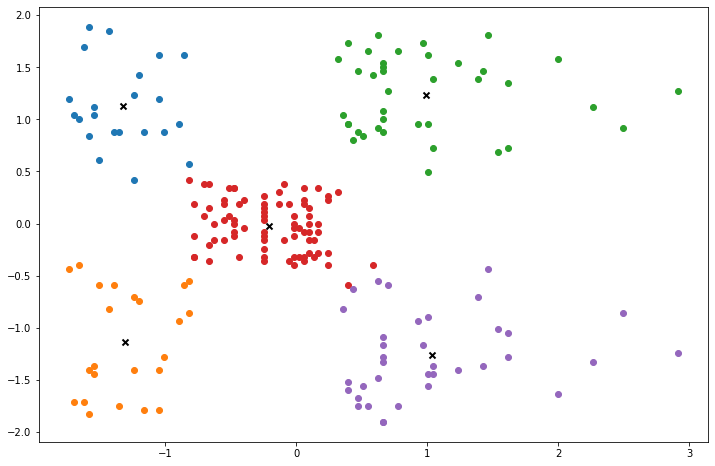
 همانطور که در تصویر مشاهده میکنیم، در k=8 به بعد تقریبا خطا تغییری نمی­کند و می­توان گفت به نقطه elbow رسیدیم البته برای پیدا کردن آن نقطه نیز باید از الگوریتم خاصی استفاده کنیم که چون در دستور کار گفته نشده بود از زدن ان پرهیز کردیم و صرفا به بیان نقطه elbow اکتفا می­کنیم.

۵ – دیتا ست Mall-Customer

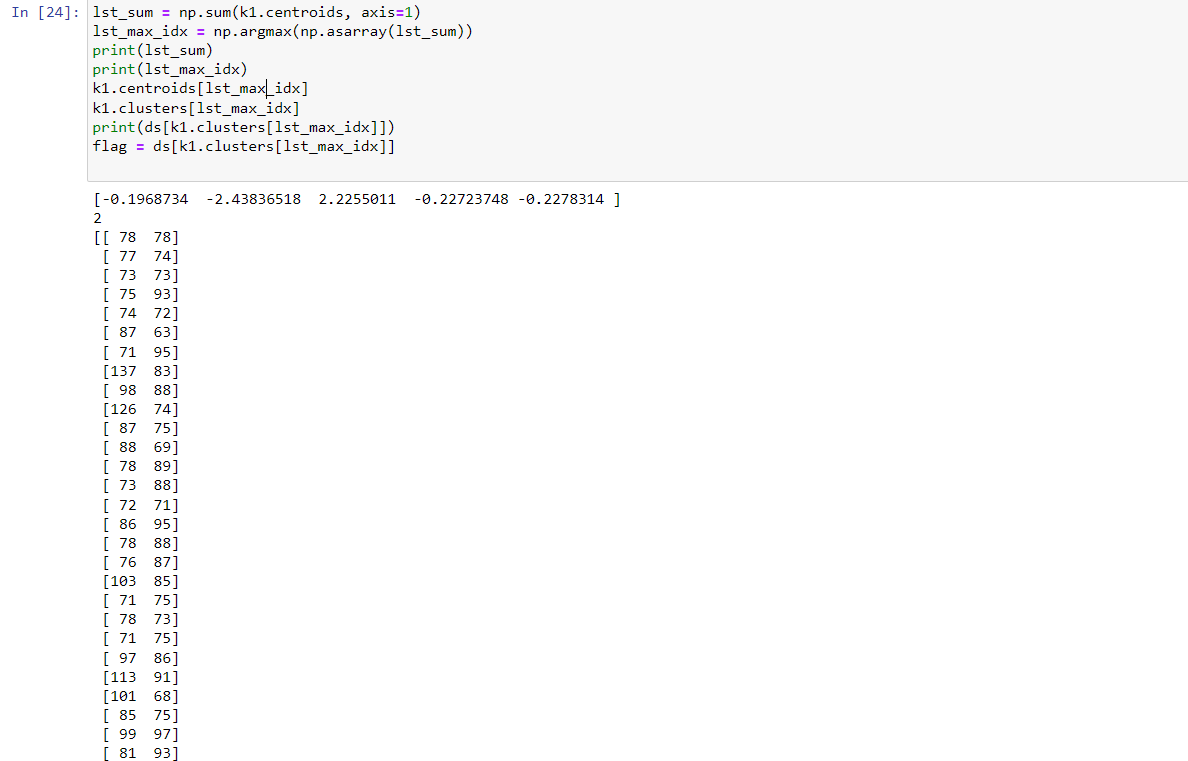
ابتدا این دیتاست را بصورت یک دیتافریم می­خوانیم و برای چک کردن اینکه کدام مشتری بیشتر راغب به خریدن کالا است را پیدا کنیم فقط نیاز به درامد سالانه و spending score مشتری داریم به این صورت که در نظر ما مشتری ای را می­توان به خرید محصول ترغیب کرد که درامد بالایی داشته باشد و احتمال خرید آن فرد نیز بالا باشد زیرا اگر مشتری درامد کم داشته باشد ولی راغب به خرید هم باشد احتمال خرید آن مشتری کم است و بالعکس. پس مقادیر ۲ ستون مربوطه را در یک دیتافریم قرار می­دهیم که از آنها استفاده کنیم.

بعد از خواندن دیتاست و شافل کردن آن، آنر نورمالایز میکنیم که روش نرمالایز کردن را در تمارین قبلی توضیح داده ایم از تکرار مکررات پرهیز میکنیم.

سپس دیتا نرمالایز شده را با یک کلاس KMeans با K برابر ۵ fit می­کنیم که خروجی را در تصویر زیر مشاهده میکنیم.

شکل۱۹) دسته های مربوط به دیتاست customer-mall

همانطور که گفته شد مشتری احتمال خرید دارد که درامد و spending-score اش بالا باشد، که در اینجا ما مشتری های دسته سبز رنگ مدنظر ما هستند حال برای اینکه دسته مورد نظر را مشخص کنیم بدین شکل عمل می­کنیم. چون دیتا ها نرمالایز شده اند ما میگوییم هر دسته ای که مرکز آن مجموع درامد و احتمال خریدش از باقی مراکز بیشتر باشد آن کلاستر مد نظر ماست. پس ما یک لیست به اسم lst\_sum تشکیل میدهیم که لیستی از مجموع درامد و احتمال خرید هر یک از مراکز است سپس ما ایندکس بزرگترین مجموع را با استفاده از تابع argmax بدست می­آوریم که این ایندکس همان مرکز مدنظر ماست سپس در متغیر flag آن کلاستر مربوط به مرکزی که مشتریان راغب به خرید را تشکیل می­دهند، ذخیره می­کنیم. در انتها نیز آن کلاستر را پلات می­کنیم.

شکل ۲۰) تصویر مربوط به پیدا کردن مشتریان مدنظر

Chart, scatter chart

Description automatically generatedشکل ۲۱) مشتریانی که احتمال خرید بالایی دارند که همان دسته سبز رنگ پلات قبلی هستند